

UC Davis

UC Davis Electronic Theses and Dissertations

Title

Visual Analytics and Machine Learning Interpretability Methods for Model Understanding, Evaluation, and Improvement

Permalink

<https://escholarship.org/uc/item/9g18k0t6>

Author

Li, Yiran

Publication Date

2024

Peer reviewed|Thesis/dissertation

Visual Analytics and Machine Learning Interpretability Methods for
Model Understanding, Evaluation, and Improvement

By

YIRAN LI
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Kwan-Liu Ma, Chair

Hamed Pirsiavash

Junpeng Wang

Committee in Charge

2024

CONTENTS

Abstract	v
Acknowledgments	vii
1 Introduction	1
1.1 Background	2
1.1.1 Machine Learning Interpretability	2
1.1.2 Visual Analytics for Machine Learning	3
1.2 Content Overview	7
2 VA and ML Interpretability for Model Understanding	9
2.1 WDS Understanding	10
2.1.1 Background and Related Work	11
2.1.2 Design Requirements	13
2.1.3 Visualization Design	14
2.1.4 Case Study	18
2.1.5 Expert Review	19
2.1.6 Discussion	20
2.1.7 Conclusion	21
2.2 ViT Understanding	21
2.2.1 Related Work	24
2.2.2 Background	25
2.2.3 Requirements and Solution Overview	28
2.2.4 Methodology and Visualization System	29
2.2.5 Case Study and Experts' Feedback	37
2.2.6 Discussion, Limitations, and Future Work	47
2.2.7 Conclusion	48
3 VA and ML Interpretability for Model Evaluation	50
3.1 Evaluation and Comparison of Model Trustworthiness	51
3.1.1 Related Work	53
3.1.2 Analysis Questions (AQs)	55

3.1.3	Methodology	56
3.1.4	Visual Analytics System	60
3.1.5	Case Study	65
3.1.6	Discussion	69
3.1.7	Conclusions	74
3.2	Evaluation and Diagnosis of Model Weaknesses	74
3.2.1	Related Work	76
3.2.2	Background and Motivation	78
3.2.3	Requirement Analysis	81
3.2.4	Visual Analytics Methodology	82
3.2.5	Case Studies	90
3.2.6	Domain Experts' Feedback	99
3.2.7	Discussion, Limitations, and Future Work	101
3.2.8	Conclusion	102
4	VA and ML Interpretability for Model Improvement	104
4.1	VA for VLM Enhancement	105
4.1.1	Related Work	108
4.1.2	Background (BLIP and SAM)	110
4.1.3	Goals and Tasks	111
4.1.4	Methodology and Visual Analytics System	114
4.1.5	Case Study, Findings, and Experts' Feedback	121
4.1.6	Discussion, Limitations, and Future Work	130
4.1.7	Conclusion	131
5	Conclusion	132
A	Appendix for Section 2.2	134
A.1	Results From Other ViT Models	134
A.2	More Details on Head Attention Patterns	136
A.2.1	Pattern Examples	137
A.2.2	Special Attention Patterns	138
A.3	Image Context Visualization	140

A.4	The Necessity of Layer-level Metrics	141
A.5	Mean Attention Distance vs. Attention Strength	144
A.5.1	Algorithm Description	145
A.5.2	Method Comparison	149
A.6	Comparing Dimensionality Reduction Methods	152
A.7	Explorations with the CIFAR10 Dataset	153
B	Appendix for Chapter 4	155
B.1	More Results on Image Data Exploration	155
B.1.1	Exploring Different Species in ImageNet	155
B.1.2	Potential Biases with Human-Related Objects in COCO	158
B.1.3	Exploring Different Scenes in Places365	160
B.2	More Results of Caption Coverage and Steering	164
B.3	Batch Processing Results of Caption Steering	165
B.4	Grad-CAM Evaluation and Computation Details	167
B.4.1	Cross-Attention Evaluation	167
B.4.2	The Computation Details of Grad-CAM	168

ABSTRACT

Visual Analytics and Machine Learning Interpretability Methods for Model Understanding, Evaluation, and Improvement

Machine learning (ML) has become indispensable in modern society, driving advancements across fields from natural sciences to personal applications. However, with these advancements comes an urgent need to understand complex model behaviors and to ensure safety and reliability in ML systems. This need gives rise to challenges in three core research topics: model understanding, evaluation, and improvement. ***First***, the increasing complexity of model architectures and the vast scale of training datasets make it difficult to gain insights into how models operate, hindering comprehensive model understanding. ***Second***, aggregated performance metrics often overlook nuanced behaviors and fail to address application-specific requirements, obscuring reliable model evaluation. ***Finally***, model improvement remains largely trial-and-error-based, with limited guidance on effective pathways for refinement. Addressing these challenges calls for new methods that not only deepen our understanding of model behavior but also enhance the safety and reliability of ML systems in real-world applications.

This dissertation addresses these challenges by creating comprehensive solutions integrating ML Interpretability with visual analytics (VA). Using interpretability techniques, I pinpoint key features and structural patterns within models, revealing insights that clarify model behavior, enhancing more in-depth model understanding. I further develop interactive VA systems that not only transform complex model interpretations into user-friendly visual representations, enabling users to intuitively understand and evaluate model behavior across various data conditions, but also provide clear guidance for model improvement.

First, I create VA and ML interpretability solutions to enhance model understanding across applications. Focusing on the understanding of a water distribution system’s simulations, I use interpretable decision trees to visualize the distributions of simulation variables and summarize optimization rules. To understand the internal mechanisms of vision transformers, I analyze the multi-head self-attention weights across different layers and heads using VA, revealing key patterns and insights in their operation. This solutions offer actionable insights, building a foundation for model evaluation and improvement.

Second, I use interpretability-driven visualizations to evaluate model reliability and robustness. Centered on assessing the trustworthiness of ML models for medical data prediction, I use VA to examine and compare the consistency of feature attributions and their value changes across different models. Meanwhile, to measure and diagnose the vulnerability of a convolutional neural network (CNN) to adversarial attacks, I design an interactive VA system to rank and explore neurons based on interpretations of their receptive fields. These efforts aim to reveal and quantify the reliability and robustness of ML models across diverse applications.

Finally, building on a deep understanding of a model’s inner workings and a comprehensive evaluation of its quality, I develop interactive systems to guide and enhance model performance. Targeting on improving image caption generation in vision-language models, I leverage VA to examine cross-attention weights linking captions to image features, uncovering gaps in feature coverage. Using these insights, interactive visualizations enable dynamic caption refinement through targeted input adjustments.

In summary, this dissertation presents solutions that integrate ML interpretability and VA techniques to advance model understanding, evaluation, and improvement across various domains. Through effective, scalable VA systems with advanced interpretability methods, these solutions systematically address key challenges in a user-centered and flexible manner, adapting to the unique requirements of different domains and model types.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Kwan-Liu Ma, for his unwavering support, exceptional guidance, and trust throughout my PhD journey. His mentorship allowed me to explore my research interests freely and persevere during challenging times, especially the pandemic. I feel fortunate to have worked under his guidance.

I extend heartfelt thanks to my mentors, Dr. Takanori Fujiwara and Dr. Junpeng Wang, who played pivotal roles in shaping my research career. Takanori guided me during my early years, teaching me the principles of proper research, while Junpeng introduced me to deep learning and machine learning interpretability, the foundation of my work. I am also deeply grateful to Dr. Xumeng Wang and Shenyu Xu for introducing me to diverse research topics and serving as inspiring role models during my formative years.

I would like to acknowledge my committee members: Prof. Xin Liu and Prof. Frank J. Loge, who provided valuable feedback during my qualifying exam, and Prof. Hamed Pirsavash, who offered insightful advice during my dissertation process. Their guidance greatly enriched my academic journey.

My sincere thanks go to my collaborators, including Dr. Erin Musabandesu, Prof. Katherine Kim, Dr. Yong Choi, Prof. Dongyu Liu, and my colleagues at Visa during internship. Their idea brainstorming and cross-disciplinary insights were invaluable, and I am grateful for the privilege of working alongside them.

I also wish to thank my labmates and friends, especially Xiwei Xuan and Hsiao-Ying Lu, for their camaraderie and shared enthusiasm for research. Their willingness to engage in spontaneous discussions and support during difficult moments meant a great deal.

A special note of thanks goes to my beloved husband and labmate, Dr. Qi Wu, whose unconditional support, patience, and belief in me have been my greatest source of strength and motivation. Your love and encouragement made all the difference in both my personal and professional life.

Finally, to my family—my parents and grandmother—thank you for your unwavering belief in me throughout this long journey. Your presence has been a constant source of comfort and inspiration. I dedicate this dissertation to my late grandfather, who always encouraged me to reach for the skies. Though he could not see the completion of this journey, I hope I made him proud.

Chapter 1

Introduction

Machine learning (ML) is advancing at an unprecedented pace, transforming fields from scientific research to daily life in ways that were once unimaginable. From medical diagnosis [1, 2] and climate modeling [3, 4] to personalized recommendations [5, 6] and autonomous systems [7], ML now underpins many essential aspects of modern society. Figure 1.1 presents an iterative ML research workflow with five main steps from data and model preparation to model deployment. As ML models grow in scale and complexity, critical challenges in applying this research workflow revolve around gaining a thorough understanding of model behavior and ensuring their safe, reliable use [8, 9].

Specifically, these challenges can be categorized into three perspectives. **First**, the complexity of ML model structures and the enormous number of parameters limit a deep understanding of the model mechanisms. Consequently, model preparation and training often rely on trial and error rather than solid comprehension, while the lack of transparency in deployed models hinders user trust. **Second**, aggregated performance metrics alone are not enough to comprehensively evaluate a model’s quality and reliability. Moreover, what defines a “good” ML model varies by user and domain, a nuance that automated metrics cannot fully capture. **Finally**, model improvement also demands extensive trial and error, as researchers lack clear guidance on potential refinement directions.

To address these challenges, this dissertation presents integrated solutions that combine visual analytics (VA) and Machine Learning Interpretability techniques. These solutions employ both existing and newly developed interpretability methods to elucidate complex model mechanisms by identifying influential features, critical components, or representative data points [10]. These methods deepen model understanding, enable more informed model selection, and en-

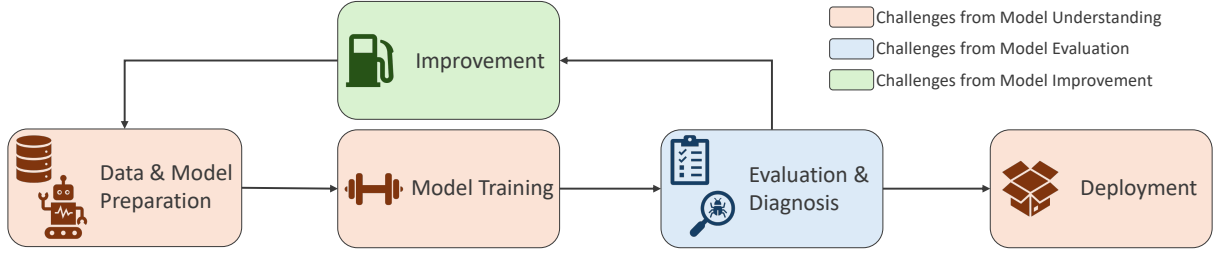


Figure 1.1: A typical workflow of machine learning research. Essentially, the workflow consists of data and model preparation, model training, evaluation and diagnosis, model improvement, and model deployment. Research goes back to the beginning of the workflow with proper model improvement if model evaluation is not satisfactory. As models grow in complexity and standards for model quality become more diverse, these research steps face challenges in understanding, evaluating, and improving ML models.

hance trustworthiness. Meanwhile, I design interactive VA interfaces that merge intuitive visualizations with automated techniques to support complex data analysis, facilitate reasoning, and uncover actionable insights [11]. Through interactive explorations and in-depth analysis, the VA systems enrich model evaluation by revealing nuanced patterns and potential issues that may go unnoticed in automated assessments. Together, the comprehensive understanding and robust evaluation enabled by these approaches provide user-centric, flexible pathways for targeted model improvements.

1.1 Background

1.1.1 Machine Learning Interpretability

ML interpretability refers to the ability to understand and explain the decisions or predictions made by ML models, making their behaviors more transparent and trustworthy [12, 10]. With the increasing prevalence and complexity of ML models in everyday applications, it has become equally important to interpret the obscure structure of the models, and understanding their intangible approximations to the relationships in the real-world datasets. For example, in [13], the authors described a case of risk predictions on pneumonia patients. They discovered through interpretable rules of a prediction model that the model regards patients with a history of asthma as ones with lower risks of death, which is plausible because these patients usually receive more aggressive treatments. However, if the model’s conclusion is adopted without doubt, patients

with asthma will be regarded as ones with lower risks of death and thus treated with less care. In this case, high-performance metrics are obviously not enough for clinicians to accept the model’s predictions, which might even lead to detrimental decisions, and interpretations are necessary to evaluate the reliability of model predictions.

The techniques developed in interpretability research can be partitioned in multiple ways based on different standards. Broadly depending on what information is explained, these techniques are categorized as interpretations for *transparency* and *post-hoc interpretations* [10]. Transparency is to unveil the underlying mechanism of an ML model, and can be decomposed into *simulatability* of the entire model, interpretability of the model’s components (*decomposability*), and the interpretability of the training algorithm (*training transparency*). Post-hoc interpretations are techniques that extract additional information besides the model itself.

In [14], the authors described techniques’ categories based on three criteria of interpretability: *complexity*, *scope*, and the level of *dependency* on the used model. Based on the model’s complexity, there are ML algorithms that are developed with interpretability by nature, such as decision tree [15] and Bayesian rule list [16], and techniques that are developed to interpret highly complex models without altering their inner workings, such as feature-based explanations [17] [18] and example-based explanations [19]. For the scope of interpretations, techniques include global interpretability, which presents the patterns within a population of inputs, and local interpretability, which tries to explain the predictions on each individual example. Finally, based on dependency on the used model, interpretation techniques can be divided into model-agnostic and model-specific methods.

1.1.2 Visual Analytics for Machine Learning

Visual analytics (VA) is “the science of analytical reasoning facilitated by interactive visual interfaces” [11]. Its emphasis on integrating human cognition with computational tools makes VA particularly well-suited to addressing complex ML problems. By enabling users to dynamically explore model behaviors, diagnose issues, and refine models, VA techniques serve as a powerful bridge between domain expertise and the insights ML can provide [20]. For example, VA tools have been shown to improve the interpretability of complex ML models by visualizing feature importance, decision boundaries, or model attention mechanisms, empowering users to identify biases and make more informed decisions [21, 22]. Thus, VA enhances ML usability by fostering trust and interpretability while accelerating actionable insights, making it essential for advancing

ML applications across domains.

There are several existing surveys that summarize VA for ML from diverse perspectives. In [22], the authors described integrating VA into the ML research process as interactive model analysis, whose pipeline is to *understand*, *diagnose*, and *refine* the machine learning model. This aligns with the proposed research directions of this dissertation. In [23], the authors proposed the same pipeline of interactive model analysis, and added a *reporting* phase after the previous three phases. They also implemented a visual system that integrates existing visualization and interaction techniques for interactive machine learning. Six major goals of leveraging VA to assist ML are summarized in [24], where each goal corresponds to one part of the ML research process, such as data processing, model parameters and data features, and learning process. In their work, an ontology of visualization for machine learning (VIS4ML) is also constructed. Hohman et al. [20] listed several motivations of visualization for deep learning, including *interpretability and explainability*, *debugging and improving*, *comparing and selection*, and *educational purposes*. I summarize in the following the development of VA techniques in different ML research aspects.

1.1.2.1 Visual Analytics for ML Interpretability

As mentioned in Section 1.1.1, interpretability embodies as different forms based on different desiderata. Among previous works of VA for interpretability, there are two main directions of research: educational explanations of ML concepts to novices, and visualizing the inner workings of ML to model users and model developers [23]. In addition to these two goals, the authors of [20] also mentioned the use of visualization as a section in applied data science and AI projects to provide interpretations that qualitatively evaluate and support their claimed contributions.

VA techniques supporting interpretability of deep learning models are categorized into *point-based* and *network-based* interpretations [22]. For point-based techniques, each component of the model (e.g., a neuron, an input, etc.) is represented as a point on a 2D plane, and relationships between these components are demonstrated by the distances between points using dimensionality reduction algorithms like principal component analysis (PCA) [25] or t-SNE [26]. For example, in the work by Rauber et al. [27], learned representations of testing samples before and after training are contrasted with scatter plots. Network-based techniques emphasize the topology information of the network structure. For example, in [28], the authors adopted network-based visualization and neuron clustering techniques to scalably explain the inner workings of a convolutional neural network (CNN). In [20], the authors summarized the ML components that

can be visualized (*what* to visualize), the techniques to visualize (*how* to visualize), and the phases that can adopt visual analytics (*when* to visualize). In addition to the point-based and network-based techniques, the authors summarized three more types of techniques: *line chart* visualization of time-varying metrics, *instance-based* analysis and exploration, and *group-based* analysis and exploration.

Although most surveys in recent years have focused on VA support for interpretation of deep learning models, there are abundant works on VA for the interpretability of traditional ML models. For example, several works provided visual analytics methods for tree-based models. To comprehensively understand random forests, iForest [29] was proposed as a VA interface disclosing all decision paths among the ensemble model. Their visual analytics system focuses on model simplification and decision path extraction for a selected group of patients. Another example is TreePOD [30] which was developed for aiding decision tree selection through visually exploring candidate trees. In the scenario of fraud detection, Collaris et al. [31] performed a case study on instance-level visual reasoning of random forest. For tree boosting methods, VA examples include [32] which presented representations of clustered trees to visualize the model, and [33] which also adopted node-link diagram to visualize individual estimators within the ensemble.

1.1.2.2 Visual Analytics for Model Evaluation and Diagnosis

The primary goal of applying VA for ML evaluation is to conduct comprehensive performance analyses of ML models. Although there are many existing metrics and tools to measure the performance, it is insufficient to conclude the quality of a model’s predictions using only an aggregated measurement, and will be difficult to correlate model’s behavior with data [34]. For example, an overall accuracy value cannot represent the classifier’s performance for each class, nor can it demonstrate the distribution of classes in the dataset. Therefore, VA is needed to provide an interactive platform for exploration and analysis of model’s performance at multiple levels. Specifically for binary classifiers, ModelTracker [35] adopted a design of colored boxes as labeled examples binned along a horizontal axis of prediction scores. This tool supports analysis of the model’s performance on the overall level, and also on the level of each feature and each instance. In another work by Alsallakh et al. [36], a VA system with two coordinated views was developed to assist analysis of a classifier’s performance on the levels of each class, the prediction confidence scores, and the distribution of feature values. They designed a confusion

wheel visualization, where confusion matrices for each class are depicted as arcs along the rings of the wheel, and predictions with lower confidence are placed on the inner rings. Squares [34] was another VA tool for performance analysis, where colored bins with different patterns are used to represent instances of multiple classes with different prediction statuses. This tool supports the expansion of bins to stacked boxes for detailed explorations, and can connect performance back to data instances for further reasoning.

Besides model-agnostic performance analysis visualization, there is also research in designing VA tools for specific types of ML models. For example, to visually diagnose the evolution of the tree boosting model during training, Liu et al. [32] represented the temporal changes of confusion matrices as a flow visualization. Similarly, Wang et al. [33] designed a Sankey diagram-like visualization to represent the temporal confusion matrix of the tree boosting model, and also connected the model’s predictions with each feature. Other examples include CNNVis [28], which used a node-link diagram to visualize the connections between neuron clusters of a CNN, and provided semantics of each cluster by visualizing packs of images.

Beyond performance evaluation and diagnosis, VA can also be applied to assess an ML model regarding other criteria. For example, one major challenge that restrains the development of ML is the lack of robustness against adversarial attacks [37]. Adversaries are able to synthesize data instances that contain imperceptible perturbations to the original data, but these instances will be mispredicted by the model with high confidence. Offering intuitive and scalable representations of model components and data features, VA is necessary for analysis and diagnosis of a model’s behavior with adversarial examples. Ma et al. [38] proposed a VA system to assist users to identify vulnerabilities of a model against adversarial attacks. Using a visualization derived from the k NN graph and coordinated views, the system demonstrates how data poisoning attacks work from the perspective of the model, the data, and the features. Another direction is to analyze and explain deep learning model’s behavior with adversarial examples. For example, Liu et al. [39] tried to extract key data paths in the CNNs that are affected by adversarial attacks, and used a river-based representation to visualize the paths across different layers. Bluff [40] is another VA tool that analyzed how a CNN is affected by adversarial examples. Categorizing neurons at each layer based on their behaviors before and after adversarial attacks, the visual interface presents the categories and their connections across layers along with feature visualizations of each neuron.

1.1.2.3 Visual Analytics for Model Improvement

Much visual analytics research has been conducted to integrate a human-in-the-loop process for ML model refinement. These works include several aspects of model refinement. The *first* aspect is to interactively revise the model based on domain knowledge. For example, the Baoba-bView [41] is a tool that enabled domain experts to grow, prune, optimize, and analyze decision trees with their expert knowledge. Another example is the ProtoSteer [42], where domain experts can review and revise the prototypes so that the prototype sequence model is refined. *Secondly*, users can interactively steer the model based on insights from the visual analytics explorations. One example is VATLD [43], which first produced diagnoses of the model’s performance on traffic light detection, and then guided users to generate examples similar to the misclassified ones for the next iteration of model training. *Finally*, several works focus on improving the model’s prediction results based on human knowledge or experience. For example, El-Assady et al. [44] proposed a visual analytics framework that allows exploration and investigation of the semantic space, allowing users to readjust the semantic relations and refine the topic modeling results.

1.2 Content Overview

This dissertation is composed of five chapters, where Chapters 2, 3, and 4 respectively focus on the VA and ML Interpretability techniques for model understanding, evaluation, and improvement.

In Chapter 2, I present my research on model understanding within the civil engineering and computer vision domains. In the civil engineering domain, I focus on pattern discovery within simulation ensembles, developing a VA system that integrates interpretable ML models to enhance the understanding of water distribution system optimization. In the computer vision domain, I dive deeper into vision transformer models, creating multi-perspective interpretation techniques to elucidate their self-attention mechanisms.

Chapter 3 introduces my research on comprehensive model evaluation for medical data prediction and image classification. For medical data prediction models, I evaluate and compare their trustworthiness based on a novel concept of prediction consistency, which measures the stability of the model’s feature attributions when the small perturbations are introduced to

the data. For image classification models, I identify and diagnose their vulnerabilities against adversarial attacks through the interpretations of neuron receptive fields.

Chapter 4 describes how VA and ML interpretability are combined to enhance the generation quality of pretrained vision-language models. Beginning with an essential understanding of how generated tokens map to image features, and through interactive identification of model weaknesses, I integrate these insights to enable user-centric, real-time interactive model steering, ultimately improving model output quality.

Finally, Chapter 5 concludes with a summary of my research goals, key findings, and their implications for advancing ML interpretability and visual analytics domains.

Chapter 2

Visual Analytics and Machine Learning Interpretability for Model Understanding

Model understanding serves as the foundation of ML research, providing essential insights that drive every subsequent stage of model evaluation and refinement. It broadly encompasses a range of insights, from deciphering the inner workings of the model itself to exploring how input data influences model behavior and outputs [45]. A solid understanding of a model’s inner workings and behavior offers numerous benefits: it enables informed choices in model preparation, such as selecting appropriate input features and preprocessing techniques, guides effective architecture design and training, and enhances transparency, thereby building trust in deployed applications [14]. Furthermore, comprehensive model evaluation and subsequent improvements rely heavily on this foundational understanding. However, as model architectures become more complex and the volume of parameters and data grows, directly analyzing the model and its inputs and outputs to gain meaningful insights becomes increasingly challenging.

VA techniques and ML interpretability together address the complexity of model understanding. ML interpretability techniques are essential for identifying and extracting critical data features, influential parameters, and key model components within complex data and model spaces. By isolating these elements, interpretability provides insights into the features or structures that drive model predictions, making it easier to trace decision pathways and distinguish different model behaviors. VA techniques, in turn, excel at transforming these intricate interpretability results into intuitive, interactive visual representations, making large-scale data and complex model parameters accessible. Moreover, VA allows users to summarize patterns and trends across broad groups of inputs and parameters, facilitating a more comprehensive view

of model dynamics. Together, VA and ML interpretability form a synergistic toolset that offers granular insights into model mechanisms and input-output relationships.

Building on the strengths of VA and ML interpretability techniques, I develop VA solutions that leverage ML interpretability to enhance model understanding in both civil engineering and computer vision. In civil engineering, I focus on the optimization of water distribution systems (WDS), where massive amounts of data generated by simulations present a significant challenge for experts to analyze and derive useful patterns. ML interpretability assists in efficiently uncovering underlying patterns with intuitive logic, while visualization provides clear interpretations of these insights, facilitating a comprehensive understanding of the complex WDS simulations. In the computer vision domain, I delve into the self-attention mechanisms of vision transformer (ViT) models. Here, I create ML interpretability techniques that process and aggregate attention patterns across different heads, performing multi-level analyses on the importance of difference heads and the spatial attention distribution within individual heads. Additionally, I develop interactive VA systems that enable systematic exploration of ViT’s attention dynamics, summarizing key attention patterns and trends across heads and layers.

2.1 A Visual Analytics System for Water Distribution System Optimization

Water distribution systems (WDSs) are complex networks that deliver water from sources to customers. They include pipes, junctions, tanks, pumps, and valves. WDSs span almost one million miles in the United States and represent the vast majority of physical infrastructure for water supply [46]. The optimal WDS design and operation should minimize the capital investment and energy costs while meeting hydraulic constraints, fulfilling water demands, and satisfying pressure requirements [47].

Our research focuses on minimizing energy costs in existing WDSs. Energy costs often vary by time of day, with customers paying higher prices when the demand for energy is at its peak. WDSs can manage the timing of energy use to lower energy costs by changing pump operations. They can further alter pump operations by pumping to an elevated water storage facility during low price periods and use gravity to serve customers during peak price times.

Hydraulic modeling software such as EPANET [48] can run hydraulic simulations efficiently to facilitate the optimization of WDS operations. By examining the simulation results of different pump operation controls, researchers can better understand how these controls affect energy costs. However, as the number of pumps increases, running simulations of all control combinations to find optimal operating schemes can become intractable. EPANET only allows the user to view the results of a single simulation at a time; consequently, comparing the results of multiple simulations is challenging. If many control combinations and simulation outcomes are tested, the analysis must be done externally, and researchers may find it hard to digest and discover insights from such large data.

WDS pump operations can be more efficiently optimized using a metaheuristic simulation optimization procedure [49]. These methods use hydraulic simulations to evaluate how different operations perform, testing out different control combinations according to a given algorithm to reduce energy cost within predefined constraints. However, these methods act as a black box [50]. Little information is extracted on which control combinations are tested or how important reliability parameters vary based on different control sets.

In this work, we develop a visual analytics system for WDS researchers to help them understand how simulation results change with pump operations. The system also enables researchers to extract actionable insights for better and more transparent optimization of WDS operations. The analysis core of our system utilizes interpretable machine learning—specifically, we employ a decision tree—to identify underlying relationships between input variables and outputs from complex hydraulic simulations. Our system provides multiple visualizations that are designed to observe the simulation data from multiple aspects, such as the similarities of input variables and temporal changes of WDS status. By effectively integrating these visualizations with an interpretable ML method, our system supports the exploration of multiple sets of simulation data, extraction of rules to optimize pumping operations, and verification of the extracted rules. We demonstrate the system’s usability through a case study on a widely analyzed WDS and assess its effectiveness based on feedback from domain experts. In addition, we provide a discussion on the extensibility of our approach to other applications.

2.1.1 Background and Related Work

In this section, we describe background and related works in WDS optimization methods and visualization of simulation ensembles.

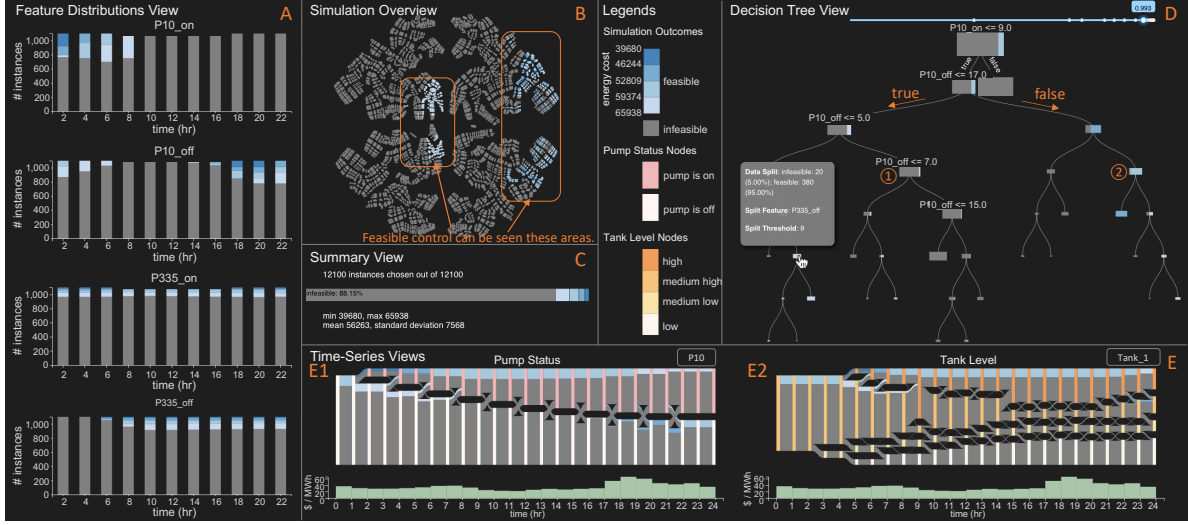


Figure 2.1: The interface of the visual analytics system for WDS optimization. Here, the NET3 simulations are analyzed. The system is composed of (A) the feature distributions view, (B) simulation overview, (C) summary view, (D) decision tree view, and (E) time-series views.

2.1.1.1 Optimization on WDS Energy Cost

There has been abundant research on optimizing operations of WDSs to minimize energy consumption [51, 52, 53]. Pump operating costs, which constitute the largest expenditure of WDSs [54], are the main focus of optimization. Explicit pump scheduling, which controls when pumps operate, is the most widely-used formulation to define this optimization problem [49]. In this chapter, we use the time when pumps are scheduled to be turned on or off as the control variables.

Hydraulic modeling software, such as EPANET [48], can be used to test out different pump control schemes to reduce energy costs while maintaining reliability. Once a model detailing the WDS is constructed, EPANET generates time-series simulations of water pressures, flows, and storage based on different operating procedures [48]. However, hydraulic modeling software typically only shows one set of simulation results at a time. This limitation makes comparing many different control schemes challenging. Metaheuristic simulation-optimization algorithms can optimize WDS operations more efficiently than the manual trial and error using EPANET. However, they do not support the comparison of many different simulation results. It is difficult to understand which combinations of controls have been tested by the algorithm and how the results of these simulations vary. Also, due to many potential control combinations and constraints, it is challenging for the algorithms to find more optimal solutions, and they may get stuck in a

local minimum [55].

Our system enables researchers to efficiently explore multiple simulation results, surpassing the capabilities of hydraulic simulations alone. It also offers deeper insights compared to traditional metaheuristic simulation-optimization approaches. The insights from this system can also be used to inform metaheuristic optimization strategies, allowing for more targeted optimization.

2.1.1.2 Visualization of Simulation Ensembles

Our work visualizes a large number of WDS simulation results and aims to extract insights from the distribution of simulation inputs and outputs. This shares the same goal with visual analytics of simulation ensembles [56]. Simulation ensembles are a collection of spatio-temporal outputs generated via computational simulations [57]. Visual analytics of simulation ensembles can be seen in various applications, such as weather prediction [58], high-energy physics [59], fluid dynamics [60], and particle hydrodynamics [61]. Throughout a comprehensive survey, Wang et al. [56] identified two common approaches of ensemble data visualization: (1) data aggregation with statistical summaries and (2) visual composition, such as juxtaposition of multiple information. Although WDS simulations usually do not have a strong focus on spatial information unlike simulation ensembles, our visual analytics system also takes these two approaches together to analyze large sets of complex WDS simulation data. Compared to existing research on simulation ensembles, which often relies on simple statistical summaries (e.g., mean and probability distributions), our approach leverages interpretable ML to uncover relationships between large sets of simulation inputs and outputs. Also, to the best of our knowledge, this study is the first research addressing a visual analytics of multiple simulation outcomes of WDSs.

2.1.2 Design Requirements

With our team’s expertise, which includes visual analytics and WDS optimization, and inspired by the analytics tasks defined for ensemble visualization [56], we identified design requirements (DRs) of a visual analytics system for WDS optimization.

- **DR1:** Provide an *overview* of the distribution of outcomes over changes in input variables. To convey general information of how input controls affect outcomes, the overview should clearly show which range of each input variable leads to infeasible and feasible controls as well as more optimal controls in case for feasible controls. An informative overview can guide researchers to filter the data to review more detailed information.

- **DR2**: Support *filtering* a subgroup of simulations to help find more insights specific to researcher’s interest. For example, the researcher may want to know why simulations within a certain range of some input variable contain a large number of infeasible controls.
- **DR3**: Extract underlying *rules* of simulations to associate inputs and simulation outcomes. The extracted rules can provide directly actionable insights to researchers for their next iteration of simulations. Furthermore, the rules can help understand the behavior of the complex simulation model.
- **DR4**: Support *verifications* and *explanations* of insights learned through the analysis. Researchers are also interested in how the extracted rules can be explained and verified by looking at the related information. Providing details of the WDS status during the simulation assists understanding the reasoning behind the rules.

2.1.3 Visualization Design

As shown in Figure 2.1, we design a visual analytics system composed of five views: (A) the *feature distributions view*, (B) *simulation overview*, (C) *summary view*, (D) *decision tree view*, and (E) *time-series views*. While (A–C) are designed for **DR1**, (D) and (E) are for **DR3** and **DR4**, respectively. To support **DR2**, each view is fully linked and is updated based on a user’s selection. Each view provides fundamental interactions, such as brushing, lasso selection, and window scrolling, as demonstrated in the supplementary video¹.

2.1.3.1 Data Description

Table 2.1 describes the structure of data we extracted from WDS simulations. The pump operation controls are formulated as time points when the pump is turned on or off and treated as inputs to the simulation model. The objective values are formulated as the energy costs obtained via a simulation plus penalties for violations of hydraulic constraints. These constraints are designed to maintain WDS reliability and include parameters, such as the minimum pressure as well as the minimum, maximum, and average tank storage levels. If there is no violation, the penalty equals zero. In this case, we categorize the corresponding input as a *feasible* solution. If there is one or more violations, a large penalty term is added to the objective value, and the value becomes incomparably larger when compared with the feasible solutions. We categorize an input that violates a constraint as an *infeasible* solution.

¹Supplementary video can be accessed through <https://youtu.be/3HB8mv0Gi28>

Table 2.1: Data structure used in the analysis of WDS simulations.

Input*				Output	
P10_on	P10_off	P335_on	P335_off	objective_value	infeasibility
2	4	2	4	4.0e+7	true
2	4	2	6	1.0e+7	true
...	...	...	...	...	...
2	14	8	6	6.0e+4	false

*Pump control decision variables/features (time of day).

Below, we describe each view by applying the system to the simulation results of a real-world WDS, called NET3 [48]. The NET3 model consists of two controllable pumps, named P10 and P335. We collected simulation results by setting time of turning on and off each pump with a grid sampling interval of two hours of the day. For example, P10_on (timing of turning on P10) is in $\{2, 4, \dots, 22\}$ and a simulation can be set to P10_on = 2, P10_off = 6, P335_on = 4, and P335_off = 6. This collection of simulations contains 12,100 simulation instances. Even though we used the NET3 model for demonstration purposes, we have tested our system with more complex WDSs, such as the Richmond skeleton system (7 pumps) [54].

2.1.3.2 Feature Distributions View

This view uses a stacked histogram to present how the simulation outcomes are distributed with changes in each individual pump control. For each pump, the simulation requires two variables (or features)—the timing of when a pump is turned *on* and *off*. Thus, Figure 2.1A consists of four histograms (2 pumps $\times$ 2 features). Each histogram’s *x*- and *y*-axes represent an input feature value (i.e., time) and frequency (i.e., the number of simulation instances). As we used grid sampling on a consistent interval, at an initial state, we have the same number of simulation instances for each feature value. This number can be changed based on filtering performed in this or other views. The color of each bar encodes the simulation output, as shown in legends in Figure 2.1. We assign a gray color for simulation instances resulting in infeasible solutions. Feasible solutions are colored with a blue sequential colormap based on their resultant objective values (energy costs) as we are interested in finding optimal pump controls.

2.1.3.3 Simulation Overview

The feature distribution view focuses on individual feature values to provide an overview of the relationships between simulation inputs and outputs. In contrast, the simulation overview

(Figure 2.1B) illustrates these relationships by highlighting the similarity between input feature values of each simulation instance. From high-dimensional data (e.g., 4 features in Figure 2.1), we embed the instance similarities into a 2D space with a dimensionality reduction method (specifically, we use t-SNE [62]). Each point represents one simulation instance and the distance between each point represents their feature value similarity. We use the same color encoding as the feature distributions view. From this view, users can see a general pattern of the relationships, such as a cluster of instances with feasible solutions.

2.1.3.4 Summary View

The summary view (Figure 2.1C) shows the selected instances’ statistical summary, including the number of selected instances, the distribution of simulation outcomes, and statistical measures (e.g., mean) of objective values with feasible solutions.

2.1.3.5 Decision Tree View

We aim to extract rules from a collection of simulation instances to identify patterns in input feature combinations. These rules indicate which combinations are likely to produce feasible or infeasible solutions, providing valuable guidance for further optimization. For this purpose, a collection of simulations can be considered as high-dimensional data with labels (i.e., feasible or infeasible). To identify features that highly influence the differences of instances with different labels, several existing visual analytics approaches utilize interpretable ML methods, such as simple neural networks [63], linear discriminant analysis [64], and contrastive learning [65, 66]. While these approaches can extract simple rules, such as values of Features X and Y highly contribute to the differences, such rules are insufficient for our case. As each pump control affects other pumps’ feasible controls, we need to extract rules that consider a combination of multiple pump controls. For example, when P10 is turned on before 9 AM and then turned off after 5 PM, simulations might judge inputs as feasible solutions no matter when turning on P335, and vice versa. To extract such interpretable rules considering multiple conditions, we utilize a decision tree (DT) algorithm [15].

The system applies a DT algorithm to the whole simulation data and then uses cost-complexity pruning [15] to make the extracted DT simple and easy-to-understand. When pruning more nodes, while the tree less precisely describes the differences between feasible and infeasible solutions, the tree becomes simpler (i.e., fewer nodes and shallower); thus, easier to interpret. A balance between the preciseness and simplicity can be controlled with a slider placed at the

top of Figure 2.1D, where the selectable prediction accuracy scores are marked with circles. These accuracy scores also present the trustworthiness of the DT representation. Furthermore, an average cross-validation score besides the slider shows the DT’s generalizability.

Similar to conventional DT visualization, we show the splitting criterion at each tree node (e.g., $P10_on \leq 9.0$). Left and right edges correspond to conditions of ‘true’ and ‘false’, respectively. We further encode essential information in each node. The size of each node’s rectangle represents the number of instances split by the node. Also, the rectangle shows the proportion of feasible and infeasible solutions with widths of inner rectangles. While we use the same color encoding with the feature distributions view, the feasible proportion is colored based on the mean of feasible solutions’ objective values. For example, the nodes annotated with ① and ② contain mostly infeasible and feasible solutions, respectively. By clicking a node, the user can select instances belonging to the node.

2.1.3.6 Time-Series Views

WDS simulation software such as EPANET can generate time-series data during each simulation, including pump statuses and tank levels at every hour. We visualize this data with Sankey diagrams [67]. In the Sankey diagrams, nodes at each hour represent pump statuses at that hour, i.e., whether it is turned on or off (Figure 2.1-E1) or water levels of a tank from high to low (Figure 2.1-E2). Relative vertical positions of nodes are also used to convey the same information. The links represent the number of instances shifting from one status/level to another status/level at the next hour. Each link also shows proportions of infeasible (grey) and feasible (blue) instances. A user can select a pump and a tank based on their interest from a drop down menu located at the top of each diagram. Pump statuses and tank levels are color-encoded with pink and orange sequential colormaps, respectively. A bar chart placed below each Sankey diagram shows the average energy cost during each hour. By correlating the width and types of links across different hours and the heights of the average energy costs, users can understand how feasibility, pump and tank statuses, and energy costs dynamically influence each other. By monitoring the proportion of instances transitioning between feasible and infeasible statuses and correlating these changes with average hourly energy costs, users can gain insights into the dynamic relationships between feasibility, pump and tank operations, and energy costs.

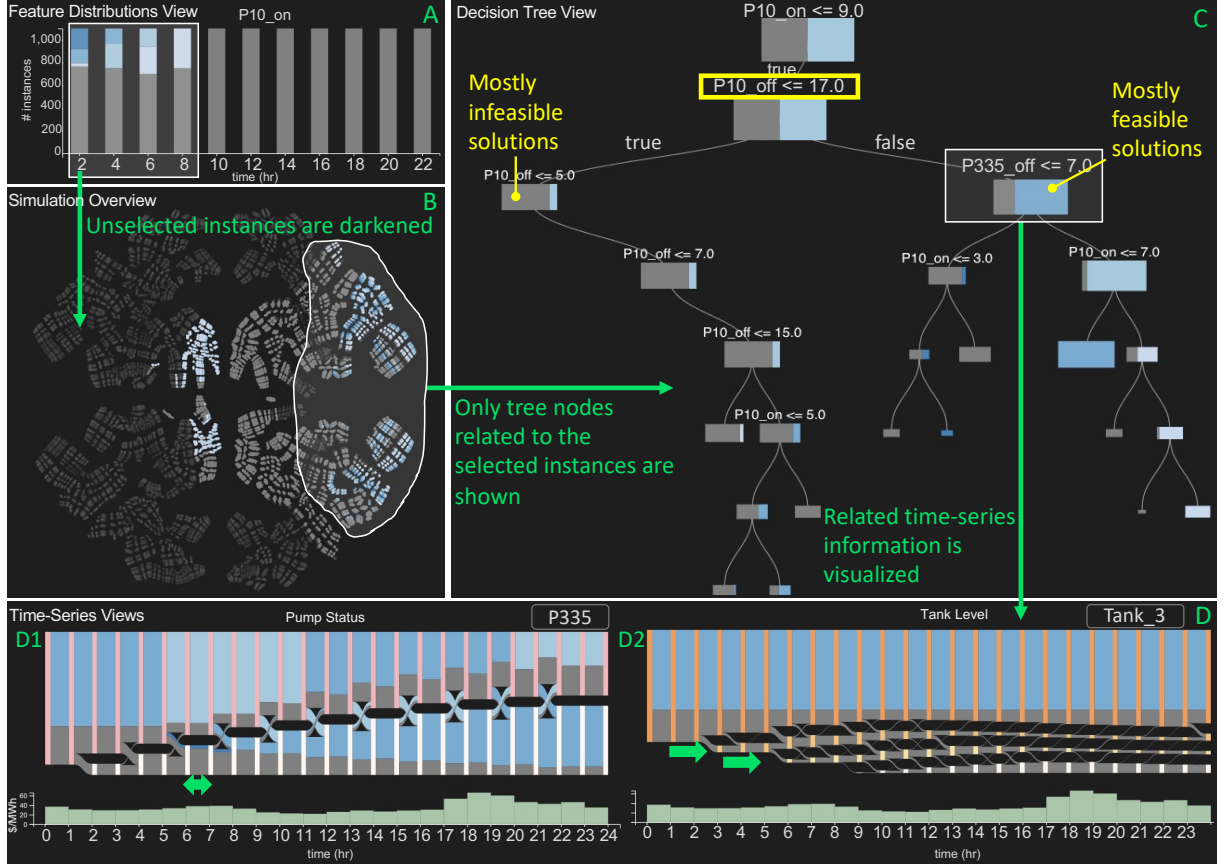


Figure 2.2: Interactive analysis performed in the case study.

2.1.4 Case Study

We demonstrate an analysis case of the simulations of the NET 3 model (refer to Section 2.1.3.1 for the details of the data) while following the required analysis processes listed in Section 2.1.2. We start by looking at the overview of the results (**DR1**). As shown in Figure 2.1, there is a much larger proportion of infeasible instances in the simulations. From Figure 2.1A, we notice that when $P10_on \geq 10$, the simulation is always infeasible. From Figure 2.1B, we see that feasible solutions are concentrated in two areas, as annotated with the orange boxes.

To examine the range containing feasible solutions in detail (**DR2**), as shown in Figure 2.2A, we perform brushing to select the area of $P10_on < 10$. From the updated simulation overview (Figure 2.2B), we can see that the selected instances are placed only at the right side of the view. Within these instances, to identify the differences between the infeasible instances and the instances having similar input features but resulting in low energy costs, we perform a lasso selection, as shown in Figure 2.2B. Identifying such differences could provide useful insights into

further optimization.

Based on the selection, the decision tree view promptly shows rules related to the selected instances (**DR3**). The results in Figure 2.2C reveal that the rule $P10_off \leq 17$ (highlighted in yellow) effectively distinguishes feasible from infeasible solutions. The left child predominantly contains infeasible solutions, while the right child mainly comprises feasible ones. We further look at the right child of the rule $P10_off \leq 17$. This child’s rule (i.e., $P335_off \leq 7$) also seems to clearly separate feasible and infeasible solutions. By combining the rules we observed so far, we can say that to achieve low energy costs, $P10_on$ should be smaller than or equals to 9, $P10_off$ should be larger than 17, and $P335_off$ should be larger than 7.

We can further understand the observed rules by using the time-series views (**DR4**). Here, as an example, we review the information related to $P335_off \leq 7$. From the pump statuses shown in Figure 2.2D1, when P335’s status becomes ‘off’ around hour six and seven (annotated with the green arrow), many simulation instances are categorized as infeasible solutions. This verifies the rule extracted from the decision tree view. Also, by looking at the tank levels of **Tank_3** shown in Figure 2.2D2, we observe that infeasible solutions tend to shift tank levels from ‘high’ to lower levels (i.e., ‘medium high’, ‘medium low’, and ‘low’). Moreover, we can see that these shifts tend to occur more frequently before or around hour seven, as indicated with the green arrows. We conclude that because of the early shutoff of P335, the tank levels become lower at an early stage of the simulation and consequently these simulations violate the constraints on tank levels.

2.1.5 Expert Review

Our visual analytics system was reviewed by three WDS optimization experts (E1, E2, and E3) to evaluate its usefulness and identify possible improvements for future study. During the interviews conducted using a video conference setup, we first explained the interface of the system and the purpose of use of each view; then, we presented a live demonstration of the case study. After the presentation, the experts were asked to provide opinions for a list of questions.

Overall, all three experts expressed that the system is a useful tool to analyze the WDS simulation data. All of them found the feature distributions view intuitive and informative. Both E1 and E2 appreciated the use of t-SNE in the simulation overview, saying that it reveals the reliability of one potential solution by viewing its “surrounding” solutions (i.e., solutions with similar input features). E3 also found the decision tree view intuitive for understanding the

separability. As for the actionable insights generated from the system, E1 said that the findings extracted from the system matched their intuition of the WDS, which would increase their confidence in use of the system. E3 also found that the functionality of filtering to see detailed information and the variability of visualization methods are helpful to generate insights.

On the difficulty of learning to use the tool, all experts confirmed that there should be a training process before using the interface in practice. E2 said that, at the beginning of the interview, they did not fully understand the usage and functionality of the system, but once going through the live demonstration of the case study, they clearly understood how to use the system for practical analyses and felt the system design is intuitive. All experts found that the Sankey diagrams in the time-series views are the least useful visualization. E1 and E3 found the Sankey diagrams hard to understand without careful explanation, while E2 thought that visualizing links used to represent changes in statuses are not necessary. Also, as a current limitation of the system, E3 mentioned that they wanted to know the breakdown of the violated constraints by infeasible solutions.

2.1.6 Discussion

Strengths and limitations. From the experts’ feedback, the most significant strength of this work is the ability to gain useful and intuitive insights from a large set of simulation data by incorporating computational analysis methods and visualizations. The simulation overview helps the qualification of an optimization solution’s sensitivity, whereas the decision tree view identifies the set of rules that can be used to inform further optimization.

However, limitations still exist in the current system, indicating future improvements. Firstly, as the Sankey diagrams used in the time-series views are not intuitive enough for the experts, we should consider using simpler visualizations in the future. Second, as pointed out by E3, some experts would prefer to analyze the simulations with more detailed information. To support such requirements, we plan to extend our system to enable flexible customization of visual encodings. As multiple selections are often needed to reach actionable insights, keeping analysis records (i.e., provenance [68]) and summarizing the applied filtering could be beneficial to achieve more efficient analyses. Also, we can expect that when analyzing larger scale of WDSs (e.g., consisting of 20 pumps) in the future, our system would suffer from the issue of visual scalability, especially in the feature distributions view. To address this issue, we can utilize the rules obtained by a decision tree to sort and filter input features based on their influence on simulation outcomes.

Finally, comparisons against other choices of visualizations and ML methods are needed to further evaluate the system usability.

Extensibility. Although our system is designed to compare multiple simulations of WDSs, the approaches taken in this work are applicable to other simulation-based optimization problems, which can be found in many real-world applications [69, 70, 71]. By showing multiple aspects of the input space and its relationship to the simulation outcomes, the visual analytics workflow can effectively provide actionable insights for better optimization. Similar to the WDS simulations, multiple runs of these simulations also generate large data consisting of simulation inputs and outcomes. The designs used in the feature distributions view, simulation overview, and decision tree view can be directly applied to other types of simulations to effectively reveal important relationships between simulation inputs and outcomes. From these relationships, insights directly usable for the next step of optimization design can be extracted, as we demonstrate in the case study.

2.1.7 Conclusion

We have presented a visual analytic system that assists in relating multiple simulations’ inputs and outcomes to gain actionable insights to optimize water distribution systems (WDSs). The system incorporates interpretable machine learning to extract general rules underlying complex WDS simulations. Our approach reveals general and detailed patterns of simulation runs from multiple aspects. This approach is applicable to many other simulations producing a pair of inputs and outcomes. Thus, our work provides tangible contributions to a wide range of optimization studies using simulations.

2.2 A Visual Analytics System for Understanding the Attention Mechanisms in Vision Transformers

Transformer models have demonstrated outstanding performance on tasks in natural language processings (NLP) [72, 73, 74] and time-series forecastings [75]. Recently, their success has also been extended to the vision domain, and the resulting vision transformer (ViT) has achieved on-par and even better performance than the state-of-the-art CNNs [76]. ViT converts a 2D image

into a 1D sequence by decomposing it into many patches and arranging the patches sequentially. Each patch is analogous to a token of sequential data, and the multi-head self-attentions are then applied to the sequence to learn the relation among tokens.

Despite the superb performance, it remains unclear how ViT works internally, especially how the multi-head self-attention works on image patches. To be specific, we found ViT designers are often puzzled by the following questions:

- *First*, how important are individual heads, and is their importance consistent across images? As different heads emphasize different pair-wise attentions, their contributions to the prediction are also different. Identifying the important ones would limit the scope of model analysis.
- *Second*, how strong is the attention between two patches that are nearby or far away from each other, and does the attention strength show any trend across layers? It is widely known that CNNs extract basic shapes/colors in early layers but complex objects/concepts in later layers. Since ViTs demonstrate on-par performance with CNNs, it becomes a natural question to ask if the models have any learning heuristic from early to later layers.
- *Third*, what attention patterns have individual heads learned, and are those patterns related to image contents? We have observed heads with interesting patterns, e.g., always attending to the patch itself regardless of the image content (content-agnostic) or only attending to patches with target objects (content-relevant). But, there lacks an exhaustive summary of all possible patterns.

Answering these questions will provide a fundamental understanding of ViTs and assist their further development.

However, there are multiple challenges in answering them. *First*, Michel and Levy [77] proved that heads are not equally important in transformers through intensive ablation studies. Hao et al. [78] proposed a self-attention attribution score for each head to quantify its importance through token interactions. These works focus on NLP tasks and use a head’s impact on the prediction to verify its importance. In ViTs, however, we find that heads with little impact on the ultimate predictions may considerably influence the intermediate representations, indicating the need to profile head importance from multiple perspectives. *Second*, for attention strengths and

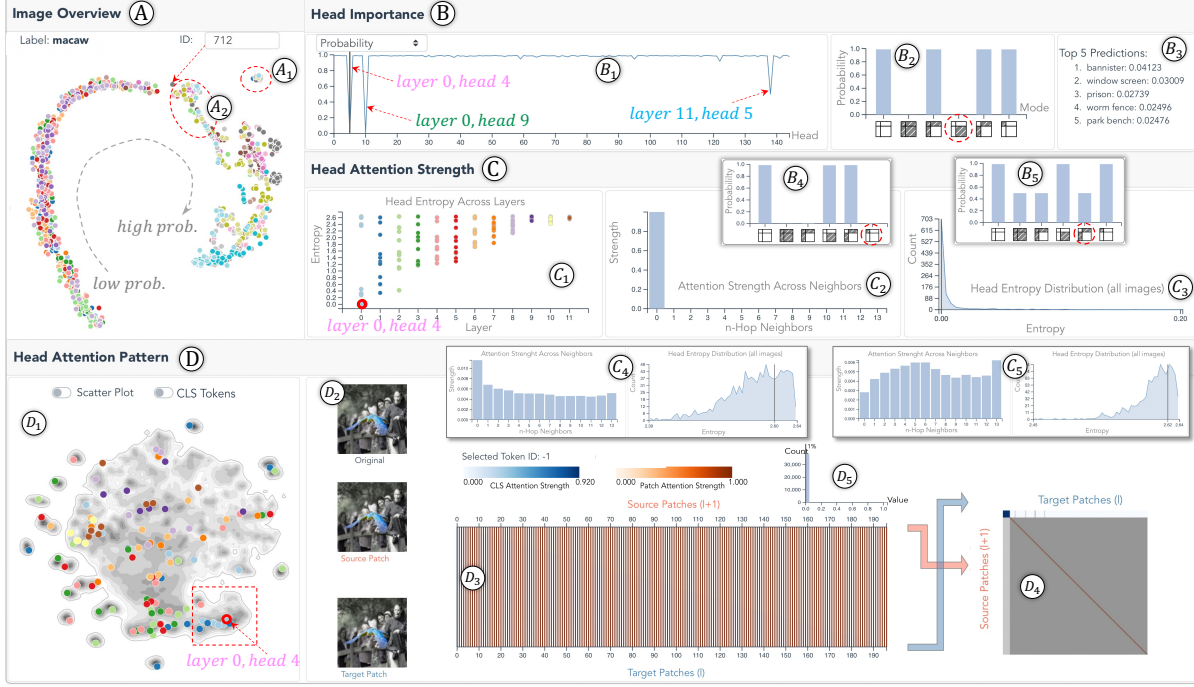


Figure 2.3: Our system contains four components. The *Image Overview* (A) lays out all images for selection. The *Head Importance View* (B) shows all heads in different importance metrics (B1) and dissects a head’s importance through partial pruning (B2, B3). The attention strengths between patches in a head are shown in the *Attention Strength View* (C), where users first obtain an overview of all heads (C1), and then focus on one head (C2, C3). The *Attention Pattern View* (D) clusters all heads by their attention pattern (D1), and presents the pattern details in one head (D2~D5).

attention patterns, multiple works have proposed to visualize them with heatmap, flow map, or matrix visualizations [79, 80, 81, 82]. However, all these works focus on language transformers with 1D attentions (forward/backward). ViTs, though rearrange image patches into 1D, still present 2D attention behaviors in the 2D spatial context (e.g., upward/downward attention). The 2D attention behavior results in much richer attention patterns, and the patches’ attention strength to their neighbors needs to be redefined considering the spatial distance in 2D. *Lastly*, based on our collaborations with domain experts, existing ViT analyses are often piece-by-piece and lack a fluent analytical workflow. For example, to understand the attention between image patches, the patches and attentions between them need to be presented intuitively and explored coordinately. However, most ViT researchers still connect the two parts manually by eyeballing them back and forth.

Our work interprets ViTs from three aspects: *head importance*, *head attention strengths*, and *head attention patterns*. For head importance, we introduce multiple pruning-based head

importance metrics, which are computed offline and can be easily plugged into our system to support head exploration and importance analysis. For attention strengths, we profile a patch’s attention strength to its k -hop neighbors as a k -dimensional vector. Aggregating the vectors from all patches of a head reflects the head’s attention strength distribution. For attention patterns, we train an autoencoder for the heads’ attention matrices, and summarize all possible attention patterns by clustering the latent representations of all heads. The three parts are integrated into a coordinated VA system. We validate the system’s efficacy by studying different ViTs with experienced deep learning experts. In short, our contributions include:

1. Multiple pruning-based metrics describing ViT heads’ importance from different perspectives.
2. A characterization of heads’ attention strength across image patches’ k -hop neighbors.
3. A comprehensive summary of the possible attention patterns in ViTs using an autoencoder-based solution.
4. An interactive visual analytics system integrating the above three parts for coordinated interpretations of ViTs.

2.2.1 Related Work

Our work belongs to the visual analytics attempts towards more interpretable deep learning (DL), with a special focus on interpreting multi-head self-attention from transformers. We thus review earlier works from these two aspects.

2.2.1.1 Visualizations for Deep Learning

A plethora of visualization works have been introduced for the interpretation of deep neural networks recently [83, 21, 84, 85, 86]. We refer readers to recent surveys [87, 88] for a thorough review of these works. Lately, deep transformers demonstrate superior performance than other DL models on 1D sequential data, and multiple visualization works have been introduced for their interpretations [79, 80, 89, 90, 81, 82]. The success of transformers has also been extended to 2D images with the seminal work of vision transformers (ViTs)[76]. However, to the best of our knowledge, no comprehensive visual analyses have been conducted to demystify this type of powerful yet complex models, especially how attention works in the 2D image context. Our work tries to fill this gap.

2.2.1.2 Attention Visualization

The attention mechanism [91] has been used extensively in DL, especially NLP-related tasks, to learn what target tokens the source tokens should “look at”. Essentially, attention is a matrix where each cell denotes the attention magnitude that the source token (row) pays to the target (column). Popular attention visualization techniques include flow maps [92, 93], parallel coordinates plots (PCPs) [89], and heatmaps [90, 80, 94]. For example, the flow maps used by Dong et al. [92] connect the source and target tokens with curves, the widths of which denote the attention strengths. Vig [89] arranges the source and target tokens along two parallel axes (i.e., a simplified PCP) and connects them with line segments in between to show the attention patterns. Heatmaps are used extensively in NLP [95, 79, 82], where the attention strengths are directly encoded into the color of each heatmap cell. There are also customized visual designs for attention visualizations [81, 90]. For instance, DeRose et al. [81] extract an “attention graph” from the attentions across layers of a BERT model and arrange the graph into a radial layout to present the propagation of attentions layer-by-layer. The resulting visualization, named Attention Flows, helps to easily analyze and compare attentions from two transformer models.

For *attention patterns* (in individual transformer heads), researchers have discovered some typical ones [80], analyzed their occurrence in different tasks [79], compared the patterns between low and high-performing models [96], and related them with the corresponding heads’ importance [82]. However, these works all focus on 1D sequential data. Attentions learned from images with a 2D spatial context have much richer patterns that are difficult to be identified and summarized manually. Our work intends to efficiently discover them. For *attention strengths* between patches within a ViT head, *mean attention distance* has been introduced in previous works [97, 76, 98], which is a sum of the attentions between patches weighted by their spatial distance. This single aggregated value holistically reflects each head’s attention strength, but also averages out many spatial details. Here, we introduce the *attention strength vector* to comprehensively profile the spatial distribution of attention strengths.

2.2.2 Background

ViT Model. The most popular ViT application is image classification, which is also the focus of this work. As shown in Figure 2.4①-⑤, a ViT classifier runs in five key steps:

1. *Decompose the input image into a sequence of patch tokens.* Without loss of generality, we

assume the same width and height for each input RGB image, denoted as w . If the patch size is $pz \times pz$, the number of patches will be $p^2 = \frac{w}{pz} \times \frac{w}{pz}$. The patches are then arranged into a sequence of tokens; each is encoded as an h -dimensional (h D) vector. Each patch token learns a concise representation for the corresponding image patch.

2. *Concatenate CLS*. A zero-initialized h D class token (CLS) is concatenated with the p^2 patch tokens, resulting in a $(1+p^2) \times h$ matrix. CLS learns to accumulate class-related features used to generate the final class probability.
3. *Add positional encodings*. The zero-initialized positional encodings are added to the $(1+p^2) \times h$ matrix. They are trained to learn each patch's positional information. We skip their details as they are not our interpretation focus.
4. *Multi-head self-attention*. This step contains l stacked attention layers, each with n attention heads. Each head learns a $(1+p^2) \times (1+p^2)$ attention weight matrix A , reflecting the pair-wise attention between all $1+p^2$ tokens.
5. *Use the CLS token for prediction*. This step decouples the CLS embedding from the patch tokens, and transforms it into class logits through fully-connected layers.

Self-Attention. Step 4 is the most important. The self-attention in each attention head (one yellow slice in Figure 2.4, right) gathers information from all $1+p^2$ tokens to learn how much attention each token should pay to itself and others. The attentions are then used to update the tokens' representations. Specifically, the $(1+p^2) \times h$ matrix at the end of Step 3, after some dropout and normalization layers, is evenly split over the n heads, each with the shape of $(1+p^2) \times \frac{h}{n}$. Inside each head, the matrix is further transformed into Q , K , and V through three separate learnable weight matrices W^Q , W^K , and W^V . The self-attention is then computed as:

$$Attention(Q, K, V) = A \cdot V = softmax(\frac{QK^T}{\sqrt{d_K}}) \cdot V. \quad (2.1)$$

The attention weight A of size $(1+p^2) \times (1+p^2)$ encodes the pair-wise attention between all $1+p^2$ tokens. For clarity, we call the $1+p^2$ tokens **source** tokens when they attend to others, but **target** tokens when they are attended by others.

Multi-Layer and Multi-Head. The self-attention computation is conducted in all n heads, and the resulting attentions are concatenated and linearly transformed to generate the final

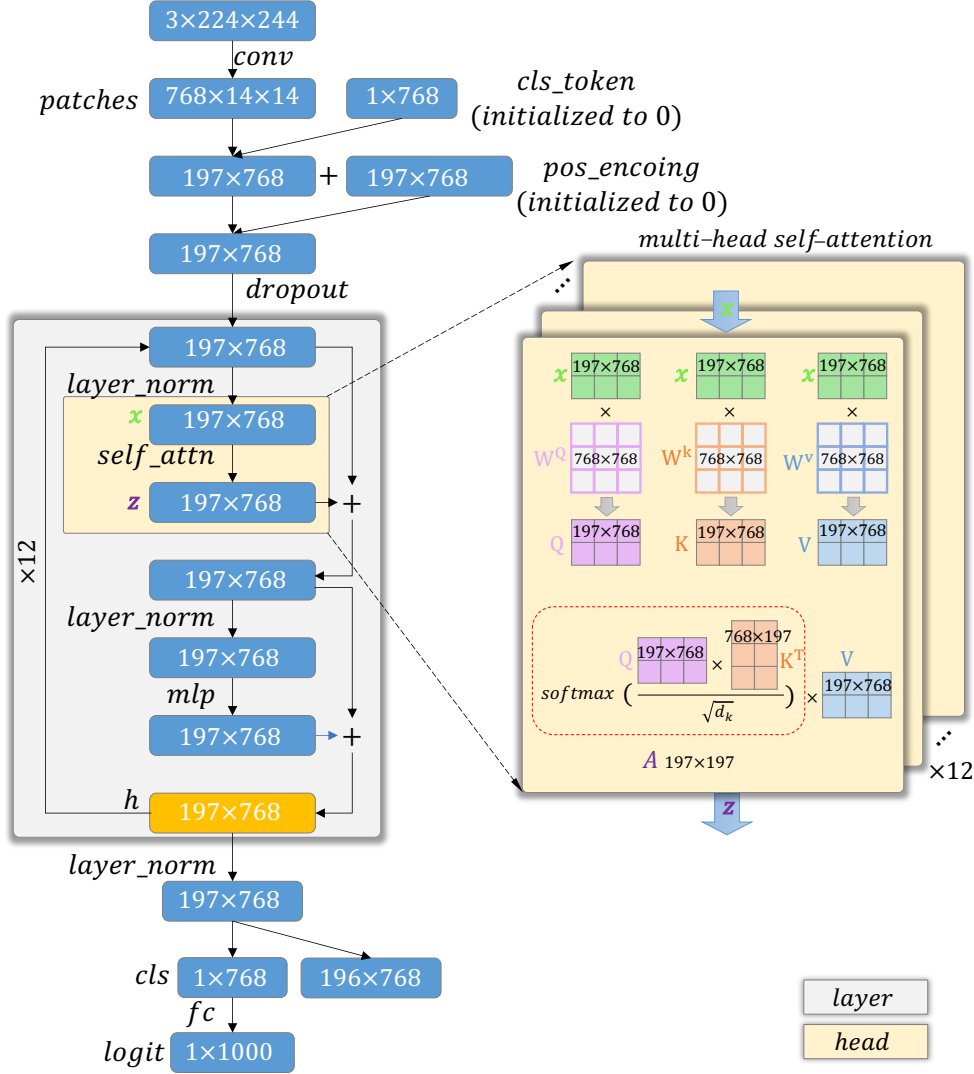


Figure 2.4: ViT executes in five steps: (1) decompose the input image into patch tokens; (2) concatenate the CLS token; (3) add the positional encoding; (4) multi-layer multi-head self-attention; (5) use CLS for prediction. Step (4) includes l attention layers, each has n heads. The attention weight in each head, i.e., A , is our interpretation focus.

multi-head self-attention, denoted as z , i.e.,

$$z = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_n) \cdot W^O + b, \quad (2.2)$$

where $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$. As shown in Figure 2.4(4), z will go through more layers to generate the final attention layer output o with shape $(1+p^2) \times h$, which is the updated hD representation for the $1+p^2$ tokens. It will be fed to a new self-attention layer, and the process is repeated for l times, resulting in l stacked layers. In total there are $l \times n$ attention heads, each with an attention weight matrix A recording the learned attention between patches in the

respective heads.

2.2.3 Requirements and Solution Overview

We maintained weekly discussions with five domain experts (all are full-time researchers with 5+ years of deep learning experience) working on transformers in vision, NLP, and time-series domains. Over these discussions and our review of the tasks in related literature [81, 80, 77, 78], we elicit the following design requirements for a visual analytics system.

- **R1: Head Importance.** To start the interpretation, we first need to quantify the importance of a large number of heads and dissect their importance. Specifically, this requires us to:
 - **R1.1:** Assess the importance of a ViT head. We want to reflect a head’s impact on both its own attention layer and the ViT’s final predictions. The impact should be assessed both on a single image and over all images.
 - **R1.2:** Dissect a head’s importance. This is to disclose the contributions from two types of tokens to a head’s importance: the CLS learns class-related features for prediction; the patch tokens learn important image contents.
 - **R1.3:** Use head importance to guide image explorations. To analyze important heads, users need to select the right images for which the corresponding heads show importance. Therefore, we need to provide an informative overview of a large number of images based on their head importance to guide their exploration.
- **R2: Head Attention Strength.** From the original ViT paper [76] and the domain experts, we noticed that most ViT designers are wondering how the patches distribute their attention strengths spatially, e.g., whether they attend more to near/far patches and how the attention strength distribution is different across heads. Thus, we should answer:
 - **R2.1:** For a single head of an image, how strong are patches attending to their spatially near/far neighbors?
 - **R2.2:** For all heads of an image, does their attention show any patterns across layers? What are the patterns?
 - **R2.3:** For a single head, does its attention strength show consistent spatial distributions across all images?
- **R3: Head Attention Pattern.** As ViT shows more and much richer attention patterns in the 2D context, it is crucial to disclose them with the image semantics. Thus, we need to:
 - **R3.1:** Exhaustively summarize all possible attention patterns from the $l \times n$ heads (for both

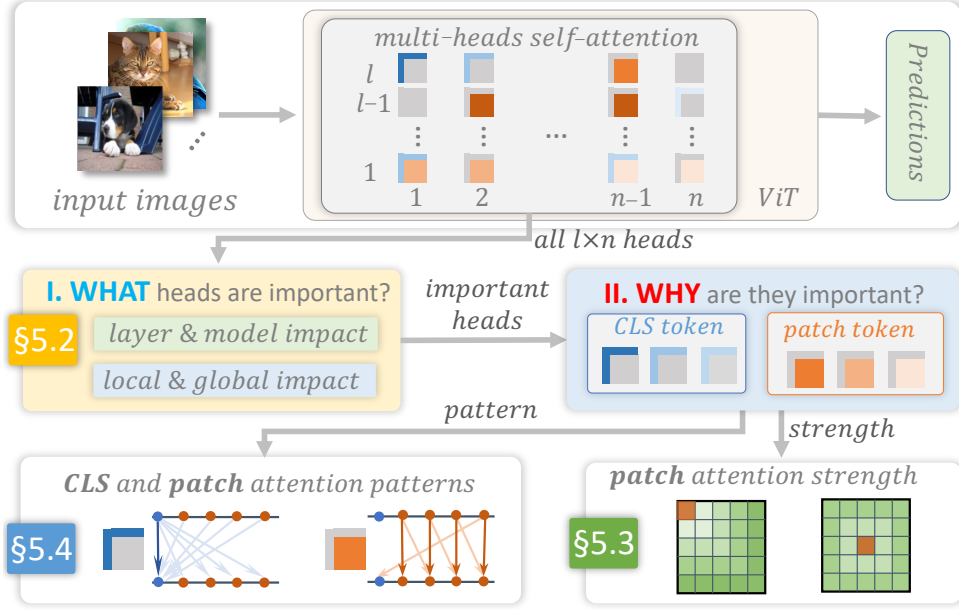


Figure 2.5: Overview of our visual interpretation solution for ViT.

CLS and patch tokens) and provide an effective overview of the patterns.

- **R3.2:** Drill down to individual heads of an image to effectively present its attention pattern and investigate if it is agnostic/relevant to the image contents.

Solution Overview. We design a visual analytics system to meet the above requirements. Figure 2.5 illustrates the system’s workflow. To interpret a well-trained ViT, we first feed the image of interest into the model to get its $l \times n$ heads. Next, we answer *what* heads are important (Figure 2.5 I) through four pruning-based metrics, meeting **R1**. Focusing on these important heads, we explain *why* they are important (Figure 2.5 II) from two perspectives. First, we disclose the attention strength distribution in individual heads by averaging attention strength across k -hop neighbors of individual patches (**R2**). Second, using an unsupervised clustering method, we summarize the attention patterns in both CLS and patch tokens and visualize the patterns in important heads (**R3**). An integrated visualization system (Figure 2.3) has been developed following the workflow.

2.2.4 Methodology and Visualization System

Our visual analytics system (Figure 2.3) contains four components: the *Image Overview*, the *Head Importance View*, the *Attention Strength View*, and the *Attention Pattern View*. The *Image Overview* (Figure 2.3A) lays out image instances based on their heads’ importance vector,

providing an entry point for the exploration. The remaining three views (Figure 2.3B-D) are designed to meet the three requirements.

2.2.4.1 The Image Overview

The *Image Overview* (Figure 2.3A) uses t-SNE+scatterplot to provide an overview of the images. Each point represents an image, and its color denotes the class label. The coordinates of each point are the dimensionality reduction result of the corresponding image’s *head importance vector*, i.e., a $(l \times n)$ -dimensional vector with the corresponding head’s importance (Eq. 2.3) at each dimension. The t-SNE layout based on this vector clusters images with similar head importance together, guiding users’ exploration (**R1.3**). For example, there is a small cluster in the top-right corner, which immediately catches users’ attention during exploration (see Section 2.2.5).

Clicking on any point or providing an ID in the top-right input box will select the corresponding image into the other three views. Inside each view, the analysis can focus on the selected image or be extended to all other images.

2.2.4.2 The Head Importance View

We define several metrics to quantify the importance of a head. These metrics are generated by “leave-one-out” ablations, i.e., encoding a head’s importance by the changes in the final output (*model-level impact*) or next-layer activations (*layer-level impact*) after pruning the head. Pruning a head is conducted by setting its attention matrix (A in Eq. 2.1) to 0. Similar head/neuron importance analysis through ablation studies has been widely adopted in NLP, e.g., [99, 77, 78].

Importance to the Model’s Output (R1.1) We propose two *model-level* importance metrics for each head. One reflects the probability change of the true class (Eq. 2.3); the other encodes the Jensen-Shannon Divergence (JSD) between the two probability distributions (Eq. 2.4) before and after a head is pruned. Mathematically, $ViT()$ denotes the well-trained model, which takes an image as input and outputs a probability distribution, i.e., $\mathbf{P} = ViT(img)$. $ViT_{i,j}()$ is the same model but the j th head from the i th layer has been pruned, and $\mathbf{P}_{i,j} = ViT_{i,j}(img)$. idx_{label} is the image’s true class index. The importance of head (i, j) is:

$$I_{i,j}^{prob} = \mathbf{P}[idx_{label}] - \mathbf{P}_{i,j}[idx_{label}] \quad (2.3)$$

$$I_{i,j}^{JSD} = JSD(\mathbf{P} || \mathbf{P}_{i,j}) \quad (2.4)$$

Importance to the Attention Layer (R1.1) Assessing only the changes in final outputs cannot reflect a head’s importance in its attention layer. As our experts noticed, pruning an

important head may significantly change the corresponding layer’s output (i.e., z in Eq. 2.2), but show minor changes to the final probabilities. This is because heads from later layers may compensate for the contribution of the pruned head, concealing its importance.

To identify the important heads in each attention layer, we propose two *layer-level* importance metrics, which are defined by the cosine distance (D_{cos}) between the immediate layer activations before and after a head is pruned. As shown in Figure 2.4, the attention layer’s output z is a $(1+p^2) \times h$ matrix, containing the activations of the CLS (the first $1 \times h$) and patch tokens (the later $p^2 \times h$). Our layer-level metrics measure the importance of the CLS and patch tokens separately. For CLS, the metric ($I_{i,j}^{CLS}$) reflects the cosine distance between the two CLS activations. For patch tokens, the metric ($I_{i,j}^{patch}$) similarly computes the cosine distances and averages the distances over all patches. Mathematically (z and z' are the layer’s output before and after pruning),

$$I_{i,j}^{CLS} = D_{cos}(z[0], z'[0]) \quad (2.5)$$

$$I_{i,j}^{patch} = \frac{1}{p^2} \sum_{i=1}^{p^2} D_{cos}(z[i], z'[i]) \quad (2.6)$$

2.2.4.3 Head Pruning Modes (R1.2)

Once the important heads are identified, we further dissect their importance by partially pruning them.

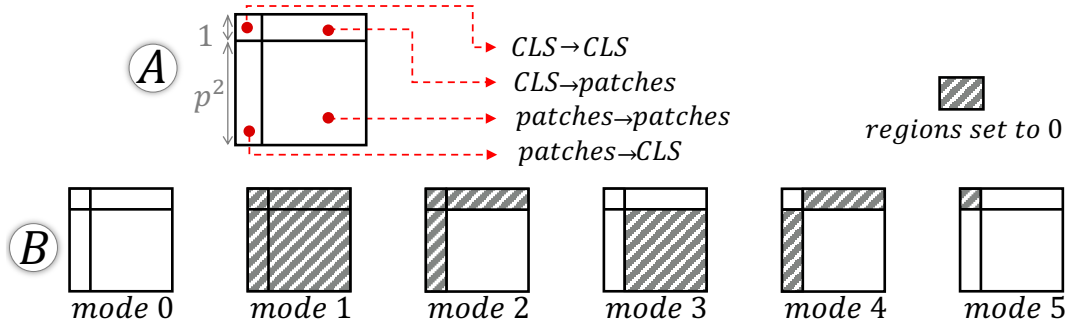


Figure 2.6: Pruning modes. (A) The attention matrix is divided into four regions. (B) Different pruning modes set one/multiple regions to 0.

As shown in Figure 2.6A, the attention matrix of a head can be divided into four regions based on the source and target tokens: $CLS \rightarrow CLS$, $CLS \rightarrow patches$, $patches \rightarrow CLS$, and $patches \rightarrow patches$. Regions $CLS \rightarrow patches$ and $patches \rightarrow CLS$ are considered together, as they both encode the interaction between the CLS and patch tokens. Six pruning modes are defined by setting different regions to zero (Figure 2.6B), i.e., mode 0 is the original head without pruning; mode 1 prunes

the head completely; modes 2~5 are additional cases where only the striped regions are pruned. Showing the impacts from these modes attributes the head’s importance to individual regions.

Visualization The *Head Importance View* (Figure 2.3B) visualizes our proposed metrics and pruning modes. First, Figure 2.3-B1 uses a line chart to present the four head importance metrics for a single selected image (i.e., the heads’ *local* importance to an image). The horizontal axis represents all the $l \times n$ heads, and the vertical axis denotes a metric’s value, where the dropdown widget enables users to switch among the four metrics. Note that for $I_{i,j}^{prob}$, we directly show the value of $\mathbf{P}_{i,j}[idx_{label}]$ (instead of the difference in Eq. 2.3) as it is more intuitive. When no image is selected (e.g., at the beginning of exploration), the curve in this view shows the average value of the selected metric over all images. Meanwhile, a blue band surrounding the curve denotes the standard deviation of the metric’s values (see Figure 2.9). The mean and standard deviation reflect the *global* importance over all images, guiding users to select globally important heads.

Second, after a head is selected from Figure 2.3-B1 (by dragging the vertical line), the bar-chart in Figure 2.3-B2 shows the selected importance metric (y-axis) in different pruning modes (x-axis), further dissecting the head’s importance. For example, Figure 2.3-B2 reveals that the importance of the selected head originates from the patch tokens solely, and pruning CLS-related attentions shows no impact.

Lastly, Figure 2.3-B3 shows the top-5 predicted probabilities for the selected image, in the current pruning. If the true label is among the top 5, it will be highlighted in bold.

2.2.4.4 The Attention Strength View

The attention strength of a head characterizes the spatial distributions of the attention strength across all patches, which answers why the head is important by disclosing where it makes the patches focus.

Attention Strength Over k -Hop Neighbors (R2.1) We define a p -dimensional (p D) *attention strength vector*, $\mathbf{s}$, for each head, which profiles the average attention strength of all patches to their k -hop neighbors ($k \in [0, p-1]$), i.e.,

$$\mathbf{s} = \frac{1}{p^2} \Sigma_i \Sigma_j \mathbf{s}^{(i,j)}, \quad i \in [0, p-1], \quad j \in [0, p-1], \quad (2.7)$$

$$\mathbf{s}^{(i,j)} = \langle s_0^{(i,j)}, s_1^{(i,j)}, s_2^{(i,j)}, \dots, s_{p-1}^{(i,j)} \rangle, \quad (2.8)$$

where $s_k^{(i,j)}$ denotes the average attention from patch (i,j) to its k th hop neighbors in the 2D domain.

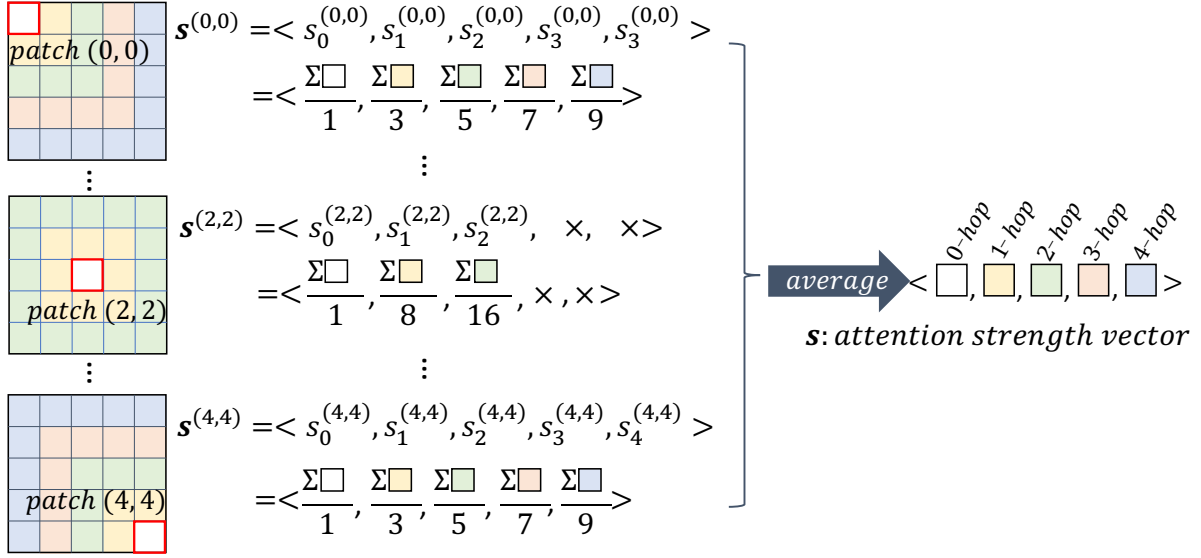


Figure 2.7: Computing the attention strength vector for a single head.

Without loss of generality, Figure 2.7 shows the computation of $\mathbf{s}$ when $p=5$. Starting from patch $(0,0)$, $s_0^{(0,0)}$ is the attention that patch $(0,0)$ paid to itself (i.e., 0-hop attention); $s_1^{(0,0)}$ is the sum of the attentions paid to its 1-hop neighbors in yellow divided by the number of neighbors (i.e., 3); $s_2^{(0,0)}$ is the total attentions paid to its 2-hop neighbors in green divided by the number of neighbors (i.e., 5); and so on so forth. To the end, we get a 5D vector for patch $(0,0)$, i.e., $\mathbf{s}^{(0,0)}$. Repeating this computation to all patches, we get 25 5D attention strength vectors, one for each patch. Their average is the head's *attention strength vector*, i.e., $\mathbf{s}$.

Note that some patches may not have certain hops of neighbors, e.g., patch $(2,2)$ in Figure 2.7 does not have 3-hop or 4-hop neighbors (marked as 'x'). Therefore, $s_3^{(2,2)}$ and $s_4^{(2,2)}$ will not be counted when computing the corresponding element of vector $\mathbf{s}$. In other words, the denominator in Eq. 2.7 is not p^2 for all elements of $\mathbf{s}$; some will have a smaller denominator due to the missing neighbors.

Visualization The *Attention Strength View* (Figure 2.3C) presents all heads' attention strength with three components. The first component (Figure 2.3-C1) presents an overview of all heads for the selected image through a scatterplot. Each point in the scatterplot is a head. Its horizontal position (as well as its color) reflects the layer that the head is from. Its vertical position denotes the entropy of the head's attention strength vector $\mathbf{s}$ (normalized). The entropy of $\mathbf{s}$

reflects if the head’s attention strength is localized on a certain-hop of neighbors (low-entropy, one element’s value dominates the vector) or spread across all k -hop neighbors (high-entropy, all elements’ values are similar). From the overview, there is an obvious trend of the heads across layers (**R2.2**), i.e., heads from higher layers attend more evenly to all patches, whereas lower-layer heads attend either locally or globally.

Second, after a head is selected by clicking the corresponding point in Figure 2.3-C1, its attention strength vector is presented as a bar chart in Figure 2.3-C2 (**R2.1**). In the current visualization, we can see that all patches in the selected head attend only to themselves (i.e., all attention strengths are distributed to the 0-hop neighbors).

Lastly, the area plot in Figure 2.3-C3 presents the distribution of entropy values for the selected head over all images (**R2.3**). For example, the currently selected head has a small entropy (Figure 2.3-C1) as all patches attend to 0-hop neighbors only (Figure 2.3-C2). Figure 2.3-C3 further reveals that the head has consistently low entropy across all images, reflected by the peak on the left corner. The vertical line over the area plot marks the head’s entropy for the currently selected image, reflecting how much the head’s attention strength for the current image varies from its strength for other images.

2.2.4.5 The Attention Pattern View

The attention pattern of a head reflects how tokens are attending to each other. We want to summarize the possible patterns of all heads to deepen the understanding of ViTs.

Unsupervised Pattern Identification (R3.1) Due to the functionality difference between the CLS and patch tokens, we treat them separately and learn their respective patterns. Specifically, given an input image and one of its heads, the corresponding attention matrix A is of shape $(1+p^2) \times (1+p^2)$ (Figure 2.4). We separate A into CLS-related attentions $A_{\text{CLS}} = \text{concat}(A[0, :], A[1 :, 0]) \in \mathbb{R}^{2p^2+1}$ and patch attentions $A_{\text{patch}} = A[1 :, 1 :] \in \mathbb{R}^{p^2 \times p^2}$, as shown in Figure 2.8A.

CLS Attention Patterns. The CLS-related attentions (A_{CLS}) concatenates the CLS→CLS, CLS→patches, and patches→CLS regions (Figure 2.8A), and its size is $(2p^2+1)$. If we have m images, each generates $l \times n$ attention matrices from the $l \times n$ heads, we will have $m \times l \times n$ such vectors. Using t-SNE, we project them from $(2p^2+1)$ D to 2D and present them with a scatterplot (Figure 2.8B). Attention heads with similar CLS attention patterns will be clustered together.

Patch Attention Patterns. We applied the same method to the patch attentions $A_{\text{patch}} = A[1 :, 1 :] \in \mathbb{R}^{p^2 \times p^2}$, but the resulting t-SNE layout could not clearly separate/cluster dissimilar/similar

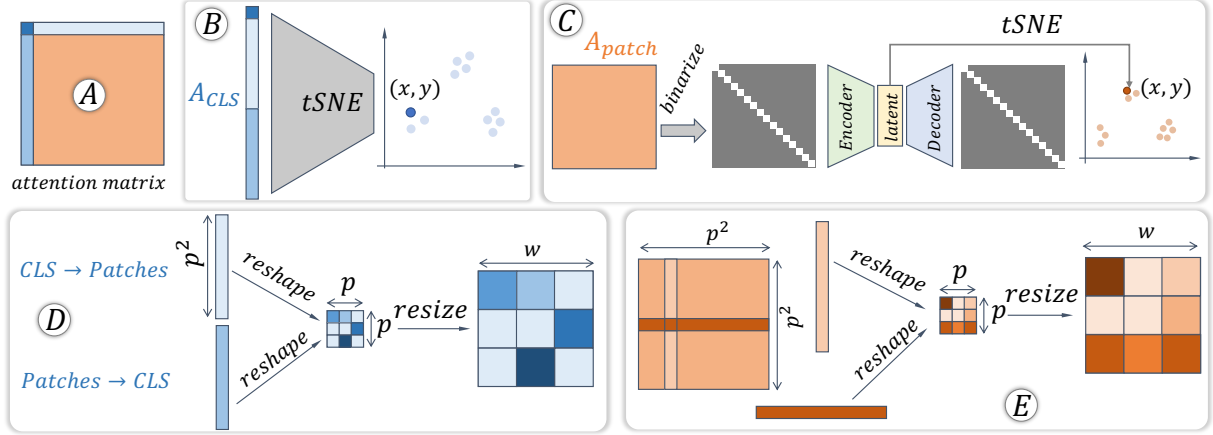


Figure 2.8: (A) Each attention matrix is separated into CLS-related attentions (A_{CLS} , blue) and patch attentions (A_{patch} , orange). (B) A_{CLS} from all images is visualized through t-SNE+scatterplot. (C) For A_{patch} from all images, we train an autoencoder to cluster them. (D, E) A_{CLS} and A_{patch} can be mapped back to the image as a mask.

patch attention patterns. We believe this is caused by the much higher dimensionality of the patch attentions and tried to fix it with several remedies. For example, we used max pooling to spatially shrink A_{patch} before t-SNE, and tried to apply PCA on A_{patch} before t-SNE. Both solutions did not yield much performance gain.

In the end, we came up with an autoencoder (AE)-based learning solution (Figure 2.8C). *First*, as we care more about the attention pattern, instead of the magnitude, we binarize A_{patch} using a cutoff, e.g., setting top 1% values to 1 and the rest to 0. This enhances the patterns and makes them easier to learn. *Second*, using the $m \times l \times n$ binarized A_{patch} , we train an AE. The AE has two symmetric subnetworks, i.e., the encoder and decoder, each with two convolutional layers and one fully-connected layer. *Third*, using the latent representations from the well-trained AE’s bottleneck layer, we conduct t-SNE layout. The layout shows obvious clusters, exposing different attention patterns.

Visualization The *Attention Pattern View* adopts the “overview+details” exploration strategy to visualize the attention patterns.

The **overview** presents all heads from all images ($m \times l \times n$ in total) through t-SNE+scatterplot (R3.1, Figure 2.3-D1), as explained in Section 2.2.4.5. The t-SNE layout could be either for the CLS attentions (A_{CLS}) or for the patch attentions (A_{patch}). The top-right toggle enables this switch. To be scalable, we allow users to convert the scatterplot into a density plot, and the top-left toggle controls this. For example, the background density contours in Figure 2.3-D1 present

the distribution of all the $m \times l \times n$ heads, as a context. When an image of interest is selected from the *Image Overview* (Figure 2.3A), its $l \times n$ heads will be shown on top of the density plot as points, the color of each reflects its layer.

The **details** of the attention matrix for a selected head are shown in the right of Figure 2.3D (**R3.2**). An attention matrix denotes the attention between $(1+p^2)$ tokens, and we present it in two different manners (Figure 2.3-D3, D4).

Figure 2.3-D3 lists all $(1+p^2)$ tokens as two rows (top: source tokens, bottom: target tokens) and uses lines with light to dark color to encode the attention magnitude. Blue and orange are used to color A_{CLS} and A_{patch} respectively. Showing all the $(1+p^2) \times (1+p^2)$ lines would make the view very cluttered. Therefore, we enable users to specify a threshold, the lines with associated attention value below which will be disabled. The histogram in Figure 2.3-D5 shows the distribution of the $(1+p^2) \times (1+p^2)$ values, guiding users to specify the threshold by dragging the vertical bar on top of the histogram. The current threshold in Figure 2.3-D5 is 1%, indicating only the top 1% lines are visible. From the dark vertical lines in Figure 2.3-D3, we can easily see that all tokens (both CLS and patch tokens) strongly attend to themselves.

Figure 2.3-D4 presents the attention matrix through a heatmap (row: source token; column: target token). The four parts of the heatmap have been illustrated in Figure 2.6A. For A_{patch} in the bottom-right corner, one pixel represents one attention value. For A_{CLS} , as one pixel is barely visible for the single row and column of attention values, we augment them to take 10 pixels. The color mapping is consistent with that in Figure 2.3-D3 (blue: A_{CLS} ; orange: A_{patch}). From the heatmap, we can observe a clear diagonal pattern, indicating that the patch tokens attend strongly to themselves. The CLS token also strongly attends to itself, as the top-left cell is in dark blue. Meanwhile, CLS also attends to different patch tokens, but the attention magnitude is very small (light blue or white color in the top row). The threshold specified from the histogram in Figure 2.3-D5 also applies to this heatmap.

The reason for presenting the attention matrix in two visualizations is to leverage their respective advantages. The heatmap shows the attention patterns more intuitively, whereas the two-axes view can better present the attention relationship between the CLS and patch tokens (one example is shown later in Figure 2.16). The two-axes view is also better than the heatmap in terms of interacting with tokens, e.g., it can easily highlight a token of interest (see Figure 2.13). Apart from these two, we have also considered other visualizations in our early design stages.

For example, we tried to overlay arrows on top of the heatmap to show the attention direction or embed patch pixels into the heatmap. However, both designs are not easily scalable to our problem size.

Image Context. To intuitively present the patch-related attentions, we need to map the patches back onto the 2D image. For $\text{CLS} \rightarrow \text{patches}$ (shape: $1 \times p^2$) and $\text{patches} \rightarrow \text{CLS}$ (shape: $p^2 \times 1$) attentions, we reshape them to a $p \times p$ square, scale the square to $w \times w$, and overlay it on top of the image as a transparency mask (Figure 2.8D, stronger attention $\rightarrow$ more transparent). For the $\text{patches} \rightarrow \text{patches}$ attentions (shape: $p^2 \times p^2$), we reshape individual row/column into a $p \times p$ square, scale it to $w \times w$, and overlay it on top of the image as a mask to show the attention from a token to all others (a row) or vice versa (a column), Figure 2.8E. The three images in Figure 2.3-D2 (top-bottom) show the original image, image+source attention mask, and image+target attention mask. Hovering over individual source/target tokens from the top/bottom axis in Figure 2.3-D3 will update the source/target attention masks dynamically (e.g., Figure 2.12D, Figure 2.13C).

2.2.5 Case Study and Experts’ Feedback

We use multiple case studies, conducted together with deep learning experts, to show the capability of our system. The experts’ feedback is summarized at the end of Section 2.2.5.5.

For ViTs, we explored four pre-trained ViTs with different image resolutions (w), numbers of layers (l) and heads (n) [76]. As our findings are consistent across them, we focus our illustrations on one model only but include results of the other three in our Appendix. The parameters of the focused model are: $w=224$, $l=12$, $n=12$, and $p=14$.

For datasets, we used 1000 images sampled from the validation set of ImageNet [100]. The images are from 20 classes (10 classes with the best and 10 classes with the worst predictions), each has 50 images. We have also explored our system with another dataset. However, as the model-level findings from the two datasets are mostly similar, we include the exploration of the other dataset in our Appendix.

2.2.5.1 Head Importance

Figure 2.3A shows the 1000 images, laid out by their $l \times n$ -dimensional head importance vector (each dimension is $\mathbf{P}_{i,j}[\text{id}_{x_{label}}]$ in Eq. 2.3). From the layout, we found images are arranged clockwise with an increasing true-class probability.

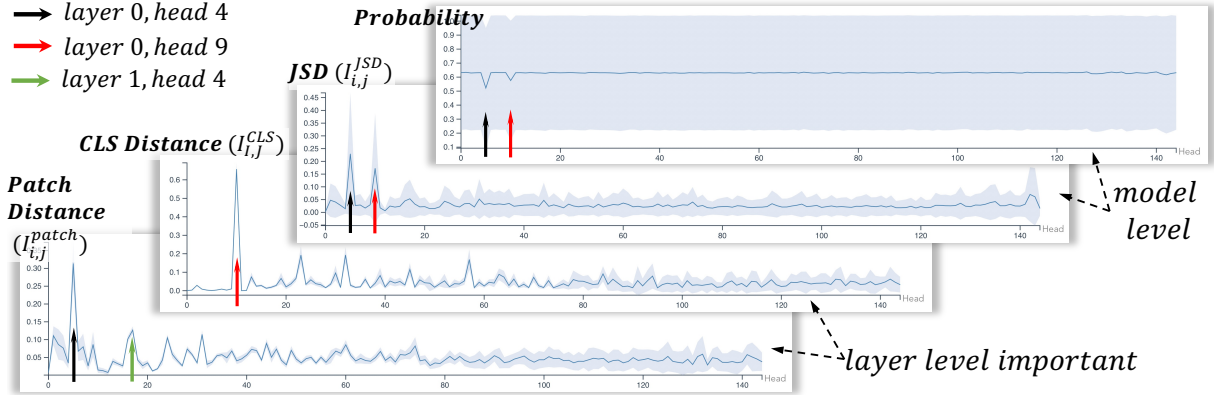


Figure 2.9: The mean (blue curve) and standard deviation (blue band) of the four head importance metrics computed over all images.

Figure 2.9 shows the four head importance metrics aggregated over all images. Both *model-level* metrics (‘Probability’, ‘JSD’) reflect head 4 and 9 from layer 0 are very important. The very wide band of the ‘Probability’ plot is due to our choice of images with the best and worst performances. The two *layer-level* metrics (‘CLS Distance’, ‘Patch Distance’) show heads 9 and 4 contribute significantly to the changes of CLS and patch representations, respectively.

Moreover, the *layer-level* metrics show more oscillations, identifying important heads that cannot be identified from the *model-level* metrics. For example, removing layer 1 head 4 barely changes the final predictions but significantly affects the layer activations. As shown in the ‘Patch Distance’ plot, the mean for this head is large and its standard deviation is small, indicating the head is important to the layer across images. This head has a fixed function of making all patches attend to the patch above themselves (explained later in Figure 2.14F). The *model-level* metrics cannot identify it as heads from later layers show similar functionalities (i.e., head 3 from layer 2, explained later in Figure 2.14J), hiding its importance. Also, an increasing standard deviation is observed from the two *layer-level* metrics, indicating that higher-layer heads’ importance is more influenced by image contents.

By coordinately exploring the *Image Overview* and *Head Importance View*, we find image clusters, to which, individual heads are very important (e.g., heads 9 and 4 from layer 0 are important to images in Figure 2.3-A1, A2). To analyze their importance, we randomly select an image (ID: 712, label: **macaw**) from one cluster for further exploration. The three heads that are very important to the selected image are: *layer 0 head 4*, *layer 0 head 9*, and *layer 11 head 5* (Figure 2.3-B1).

Figure 2.3-B2 shows the partial pruning results for *layer 0 head 4*. The probability drops only if the *patches*→*patches* attentions are pruned, and pruning CLS-related attentions has little impact. This attributes head 4’s importance to its attentions between patches. Figure 2.3-B3 shows the top-five probabilities are very low when this head is pruned, and *macaw* (the true label) is not among them.

Figure 2.3-B4 and B5 show the partial pruning results for *layer 0 head 9* and *layer 11 head 5*, respectively. From them, *layer 0 head 9* is important due to the CLS self-attention (CLS→CLS); *layer 11 head 5* is important due to the attentions between CLS and patches (CLS→*patches* and *patches*→CLS).

2.2.5.2 Attention Strength

Next, we examine why the heads are important through their attention strengths. The attention strength analysis is for patch tokens only (CLS has no spatial information), so our analysis focuses on *layer 0 head 4*. Figure 2.3-C1 shows the attention strength overview of all heads. *layer 0 head 4* (in the red circle) has the smallest entropy. Figure 2.3-C2 shows the head’s *pD k*-hop neighborhood vector. From it, all patches’ attention strengths focus on the 0-hop neighbor, i.e., the head makes all patches strongly attend to themselves. It is reasonable that such a head is important in lower layers, as no details should be overlooked at the beginning. From Figure 2.3-C3, we also notice that this head’s functionality is consistent across all images, as its entropies for different images are always low (distributed dominantly to the left).

Figure 2.3-C4, C5 show the attention strengths of the other two important heads. Their importance majorly comes from CLS-related attentions, and the attention strengths (for patch tokens) are scattered across all hops of neighbors (the bar chart) and varying across images (the area plot).

The overview in Figure 2.3-C1 and Figure 2.10 also reveals the attention strength distribution of heads over layers. In general, lower-layer heads can make patches attend strongly to their local regions, e.g., 0-hop or 1-hop neighbors (the low entropy heads in Figure 2.10A-B). Low-layer heads can also make patches evenly distribute their attention across the entire image (e.g., Figure 2.10C). For higher-layer heads, patches only attend globally with similar attention strengths across all *k*-hop neighbors, e.g., Figure 2.10D-E. A similar overview can always be observed no matter which image is selected. This observation is consistent with the original claims about attention strengths [76] (see details about [76] in our Appendix).

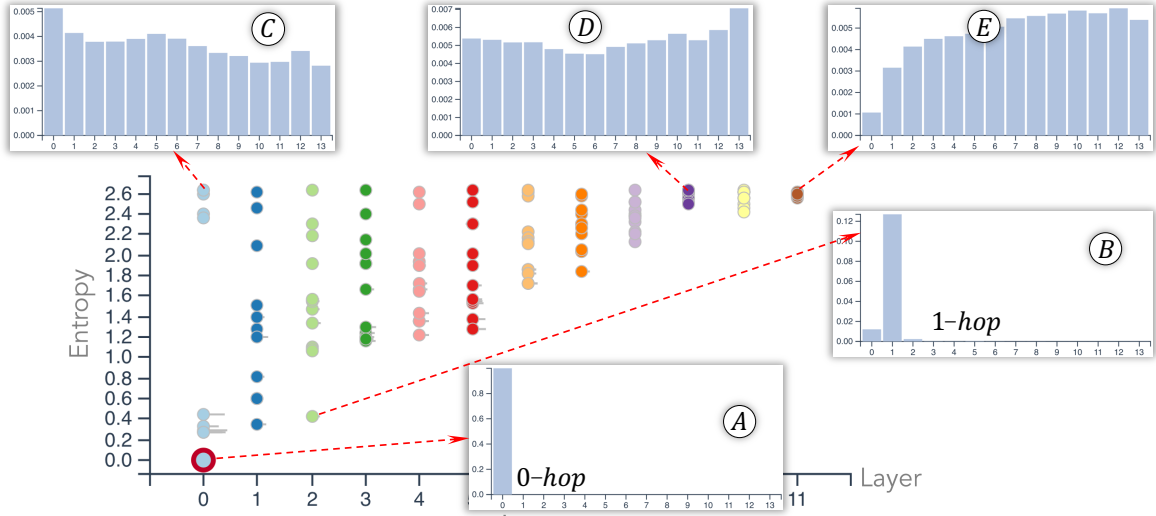


Figure 2.10: Lower-layer heads can distribute patches’ attention strength only to their near neighbors (A, B), or evenly to all k -hop neighbors (C). Higher-layer heads can only do the latter (D, E).

2.2.5.3 Attention Pattern

The attention patterns further answer why *layer 0 head 4* is important. From the *Attention Pattern View*, i.e., the vertical lines in Figure 2.3-D3 and the diagonal pattern in Figure 2.3-D4, all patches in this head strongly attend to themselves, echoing our earlier findings from the *Attention Strength View*.

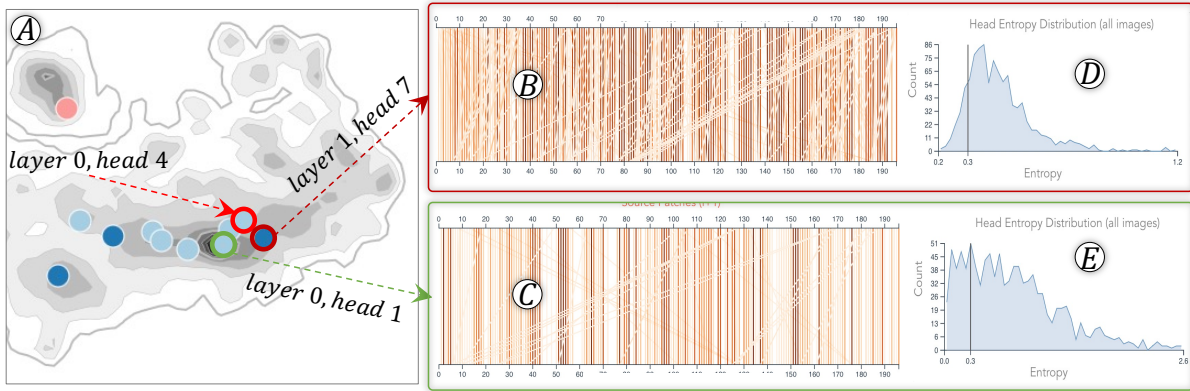


Figure 2.11: Exploring heads that have similar attention patterns with *layer 0 head 4* to explain why it is important while others are not.

Are there heads with attention patterns similar to *head 4*? If yes, why is only *head 4* so important? To answer these questions, we zoom into the red dashed region of the t-SNE layout in Figure 2.3-D1. The zoomed-in details are shown in Figure 2.11A. From it, we explore heads

close to *head 4* and check their attention patterns. Figure 2.11B-C show two of them, where the vertical lines indicate the patches also majorly attend to themselves in these two heads. However, different from *head 4*, the self-attentions in these two heads are not always strong, i.e., all lines in Figure 2.3-D3 are in dark orange, but most lines in Figure 2.11B-C are in light orange. Moreover, the two heads' functionality is not as consistent as that of *head 4* across images (comparing Figure 2.3-C3 and Figure 2.11D-E).

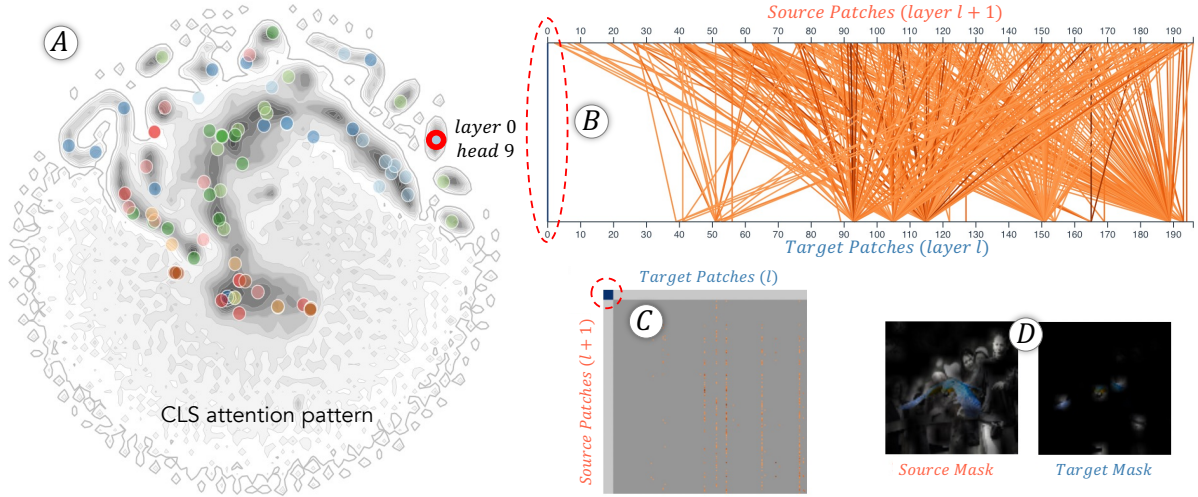


Figure 2.12: The attention pattern of *layer 0 head 9*.

For *layer 0 head 9*, we have known its importance comes from the CLS→CLS attention in Figure 2.3-B4. Visualizing its attention patterns, we can see the dark blue vertical line in Figure 2.12B and the dark blue cell in the top-left corner of Figure 2.12C, echoing the importance of CLS's self-attention. Meanwhile, the CLS→patches and patches→CLS attentions are not noticeable. The patches→patches attentions (the orange lines in Figure 2.12B and the vertical pattern in Figure 2.12C), as well as the masked source and target images (Figure 2.12D), show

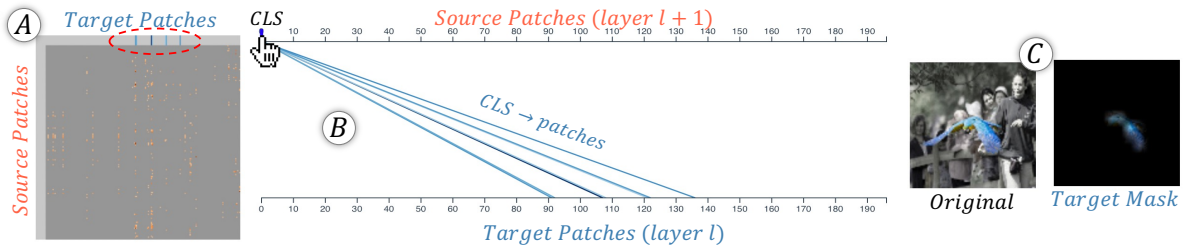


Figure 2.13: The CLS→patches attention in *layer 11 head 5*.

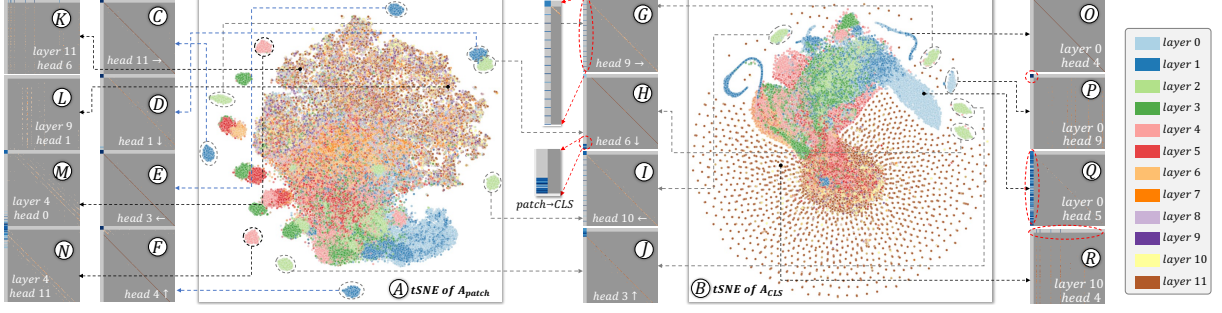


Figure 2.14: (A/B) t-SNE layouts of heads using their patch/CLS attention patterns. Content-agnostic heads from lower layers show similar patterns across images, and thus each forms an isolated cluster no matter they are laid out by the patch (C-J, M, N) or CLS (G-J, O-Q) patterns. Content-relevant heads from higher layers show dissimilar patterns for different images. These heads are scattered in both layouts (K, L, R).

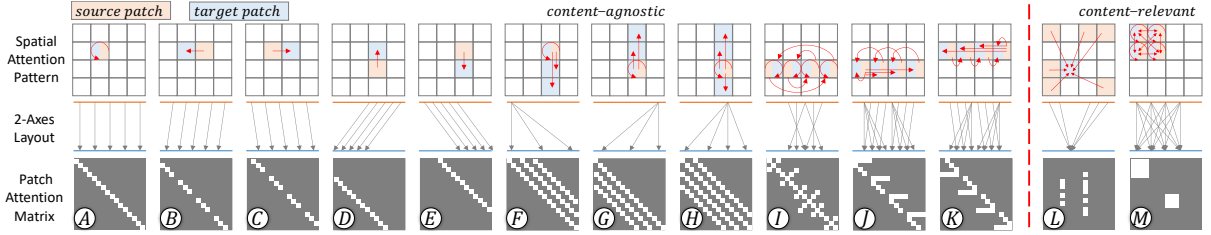


Figure 2.15: The 13 possible attention patterns between image patches presented in the (top) image space, (middle) two-axes, and (bottom) heatmap visualization. Each pattern is a mix of one/multiple of the four basic patterns: *diagonal* (A-K), *horizontal* (J, K), *vertical* (L), and *block* (M). The first two are *content-agnostic* and often appear in lower-layer heads; the latter two are *content-relevant* and often occur in higher-layer heads.

no obvious extracted features, confirming the less importance of the patch-related attentions. Also, we noticed that the CLS attention pattern of *head 9* is very unique, as it is the only head (of the selected image) in the isolated cluster of Figure 2.12A (the t-SNE layout of all heads' A_{CLS}). The background contour shows the head 9 from other images, reflecting that head 9's functionality is fixed across images.

For *layer 11 head 5*, its importance is from the *patches* $\rightarrow$ CLS and CLS $\rightarrow$ *patches* attentions (Figure 2.3-B5). In Figure 2.13A, the *patches* $\rightarrow$ CLS attentions (the leftmost column) are not noticeable, but the CLS $\rightarrow$ *patches* attentions (the top row) show four major regions, indicating the CLS attends to four groups of patches. Hovering over the CLS token on the top axis of Figure 2.13B, the four groups of target patches are highlighted in the masked target image (Figure 2.13C). From it, the four regions accurately extract the macaw's wings in blue.

Attention Pattern Summary Our explorations identified different attention patterns. This section provides an exhaustive summary.

Patch Attention (A_{Patch}) Patterns. Figure 2.14A shows the t-SNE layout of all heads’ patch attentions from the 1000 images (Figure 2.3-D1 is the density plot of it). The points from an isolated cluster often represent the same head from different images, verifying the head’s fixed functionality. Exploring individual heads in the *Attention Pattern View*, we summarize them into 13 patterns in Figure 2.15. The three rows of the illustrative figure show how each patch attends to others in the image space (top), the two-axes (middle), and the heatmap (bottom). From the last row, each of the 13 patterns is a combination of four basic patterns, i.e., *diagonal*, *horizontal*, *vertical*, and *block*.

Figure 2.15A-K include the *diagonal pattern*. Figure 2.15A denotes self-attention. Figure 2.15B-C show each patch attends to its left or right patch (one cell off the diagonal), where the gaps indicate the leftmost or rightmost patches without further ones to attend to. Each patch in Figure 2.15D-E attends to the patch above or below itself, i.e., the white-squares shift from the diagonal for a row of patches. The four heads in Figure 2.14C-F (all from layer 1) show examples where each patch attends to its right, bottom, left, and top patch, respectively. The four heads in Figure 2.14G-J (all from layer 2) show similar patterns, revealing the repeating functionalities. Coming to middle attention layers, patches can attend to multiple patches above and/or below themselves (Figure 2.15F-H). Figure 2.14M-N show two such examples from layer 4. Figure 2.15I includes the counter-diagonal patterns, in which the left/right patches symmetrically attend to the right/left ones in the same row.

Figure 2.15J-K contain the *horizontal pattern* (mixed with the diagonal pattern). The pattern indicates that a patch attends to multiple patches before/after itself in the same row.

Figure 2.15L shows the *vertical pattern*, i.e., multiple source patches (heatmap rows) attend to the same target patches (heatmap columns). This usually indicates the target patches include important semantics to the class (e.g., the **cat** face region of a **cat** image, Figure 2.14K-L). The pattern often occurs in higher-layer heads, as indicated by the yellow/brown color in Figure 2.14A (the big chaotic cluster in the center).

Figure 2.15M shows the *block pattern*, i.e., patches in a local region mutually attend to each other (e.g., the face patches of a **cat** attend to its body patches and vice versa). Similar to segmentation, the attentions extract the object’s pixels.

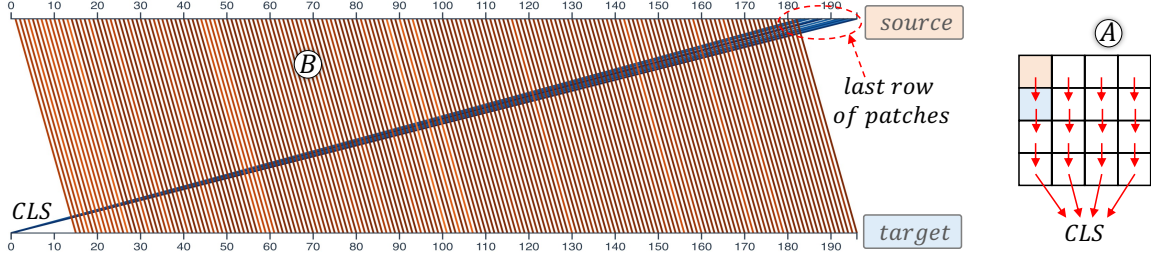


Figure 2.16: Each patch attends to the one beneath it, e.g., patch 0 (row 0, column 0) attends to patch 14 (row 1, column 0, $p=14$). The bottom row of patches attends to CLS since there are no further patches.

Attention patterns can also be categorized into *content-agnostic* and *content-relevant*. The *diagonal* and *horizontal* patterns are often agnostic to the image content, e.g., heads in Figure 2.14C-J. The same head from all images forms an isolated cluster in Figure 2.14A. The *vertical* and *block* patterns are content-relevant, as their position depends on the content of images. They are the big chaotic cluster in Figure 2.14A. Content-agnostic patterns often occur in lower layers, whereas content-relevant patterns often occur in higher layers.

CLS Attention (A_{CLS}) Patterns. The CLS-related attention patterns follow a similar layer-wise trend. As shown in Figure 2.14B (its density plot is in Figure 2.12A), heads in lower layers form clear clusters, indicating the CLS’s attentions in them are more content-agnostic. For example, Figure 2.14P shows dominantly strong CLS→CLS attentions; Figure 2.14Q shows strong patches→CLS attentions. In higher layers, the attentions are more content-relevant, e.g., in Figure 2.14R (the top row), the CLS focuses only on specific image patches.

The four heads with isolated clusters in Figure 2.14A, G-J also form four isolated clusters in their CLS layout (Figure 2.14B, G-J). By coordinately exploring the patch and CLS patterns, we found the boundary patches without further patch to attend in these heads will attend to CLS. For example, in Figure 2.14H, all patches attend to the patch beneath themselves. The patches in the bottom row have no patches beneath them, so they attend to CLS (Figure 2.16A). The two-axes view of this head is shown in Figure 2.16B, which is consistent with the pattern in Figure 2.14H (and the inset). This explains how information is passed across patches row-by-row to CLS for classification.

2.2.5.4 Head Attention Diagnosis

Our coordinated system also helps to diagnose the roles of different heads (especially the important ones) in mispredictions. We brief two example cases in this section.

Case 1: Figure 2.17A shows an image from the **overskirt** class, but the ViT performs badly on its prediction (Figure 2.17B). From the *Head Importance View*, we notice a sharp increase in the true label probability when head 9 of layer 11 is pruned (Figure 2.17C). The partial pruning result of this head (Figure 2.17D) reflects that the probability of **overskirt** will *increase* only if the attentions between CLS and patches are pruned. As the *patches*→CLS attentions are very small in Figure 2.17E (the first column), we hover over the CLS→*patches* attentions in Figure 2.17F. The masked target image (Figure 2.17G) shows that the CLS attends strongly to the background fences. Pruning such a misleading head will help the model focus more on the right features, leading to a probability increase. The importance of this head also explains why the top two classes in the original predictions (Figure 2.17B) are **fences**.

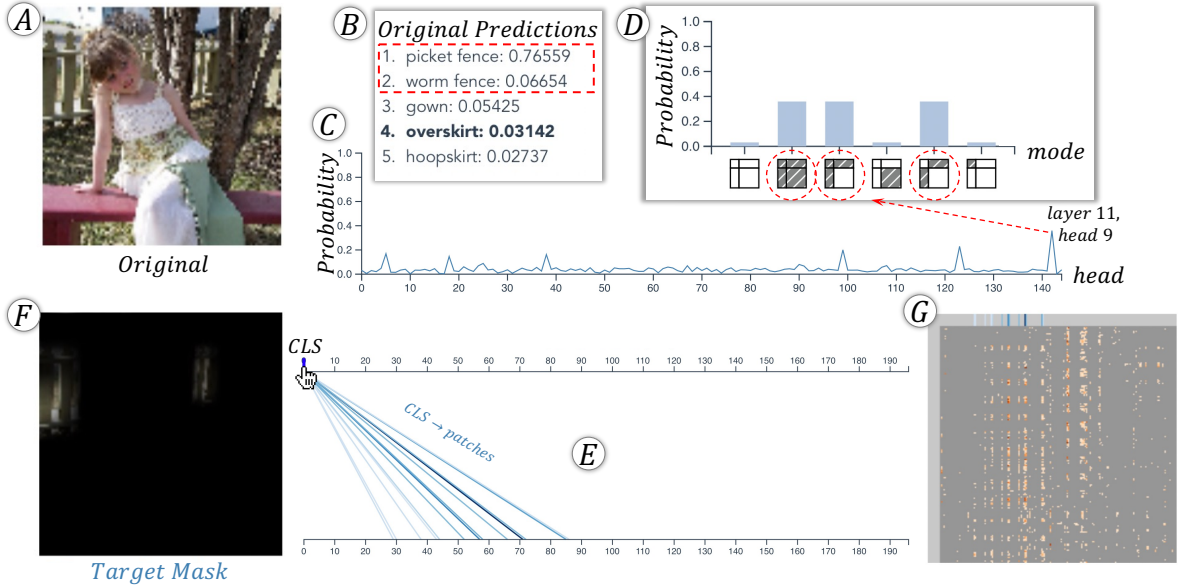


Figure 2.17: Head 9 from layer 11 incorrectly attends to the background fence. Pruning it will increase the true class's probability.

Case 2: The ViT correctly predicts the image in Figure 2.18A to be **sunglass**, but also assigns a high probability to **jersey**. We found two important heads in the last layer but with opposite effects (Figure 2.18B). Pruning heads 9 and 10 from layer 11 separately leads to a big decrease and increase of the **sunglass** probability. From the partial pruning results, we found the attentions between CLS and patches dominate the decrease (Figure 2.18C) or increase (Figure 2.18D). We then visualize the CLS→*patches* attentions and the corresponding masked images. For head 9, the CLS focuses on the face region, whereas for head 10, the CLS attends

solely to the jersey region. Heads 9 and 10 contribute largely to the probability of **sunglass** and **jersey**, respectively. Pruning them increases the opposite class's probability (Figure 2.18C, D). These details significantly deepen the understanding of how ViT works.

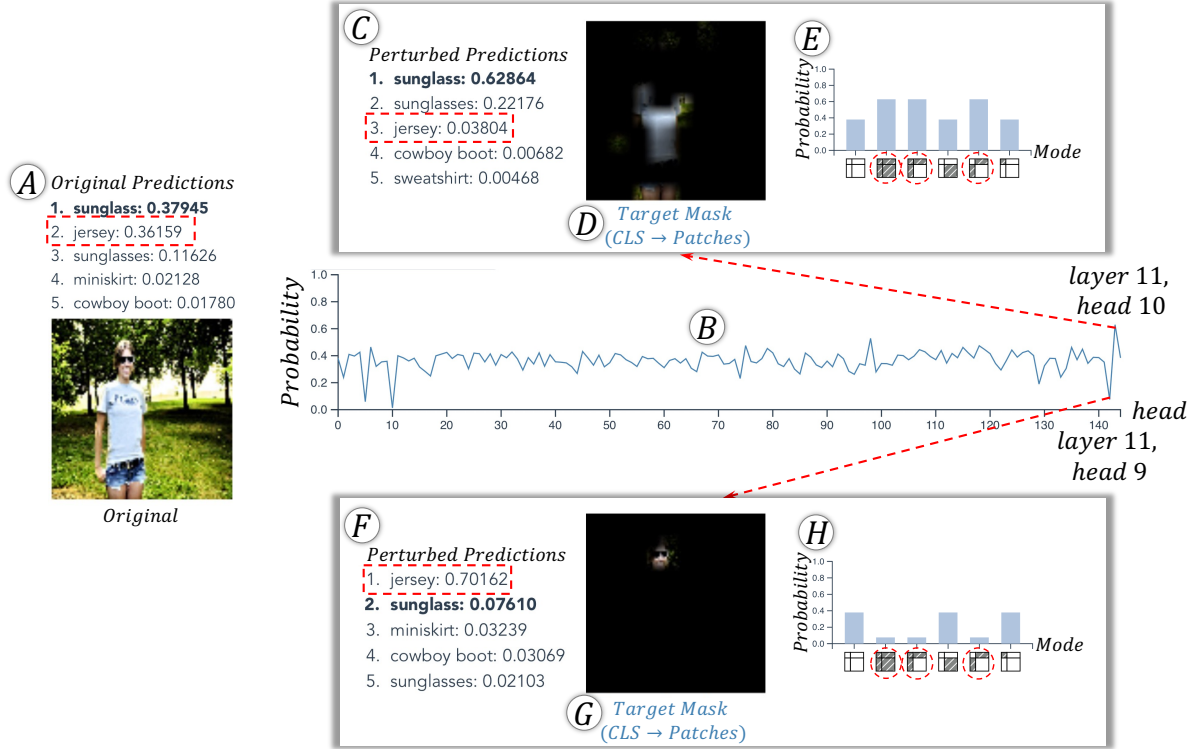


Figure 2.18: Heads 9 and 10 (layer 11) extract the **sunglass** and **jersey** features, respectively, impacting the corresponding classes' probability.

2.2.5.5 Domain Experts' Feedback

The above case studies were conducted with 7 deep learning experts in separate sessions, using the protocol of *guided exploration+think-aloud discussions*. All experts are researchers with 5+ years of experience in deep learning. Five experts ($E_1 \sim E_5$) have participated in our requirement analysis. The other two ($E_6 \sim E_7$) had no knowledge about our visualization system until the case study sessions.

In general, all experts confirmed the importance of the three focused topics and appreciated our findings. E_2 , E_4 , and E_5 enjoyed the system's interactivity, especially the linked visualizations, which helped them connect the dots for comprehensive interpretations of important heads. Their existing visualization tools with piece-by-piece analysis fall short of such coordinated explorations. Using our system, E_1 and E_3 obtained an overview of ViTs' head attention patterns

for the first time. Both experts found our findings intriguing and had thorough discussions on the analogy between CNNs and ViTs. For example, patches in Figure 2.14G always attend to their right patch. The 3×3 CNN filter $\begin{bmatrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{bmatrix}$ also aggregates the right pixel’s value to the current pixel. So, the ViT head and CNN filter share equivalent functions. Similar equivalence analysis can also be extended to other heads/filters. E_7 pointed out similar learning trends across layers of CNNs and ViTs, i.e., lower-layer heads/filters focus on local features, whereas higher layers aggregate the output of lower layers to extract object-level information. E_7 also found the interactions between the CLS and patches very insightful. In Figure 2.16A, all patches attend to the patch below them and the last row attends to the CLS. This is similar to propagating the information top-down, and the final accumulated results are passed to CLS for classification. E_6 was interested in the content-agnostic/relevant attentions and believed they could be adopted for anomaly detection.

The experts also pointed out some insufficiency of the current system. First, E_7 initially thought only the three heads in Figure 2.3-B1 were important to the prediction of image 712, which was misleading as we only pruned one head at a time and did not consider the dependency between heads. Second, both E_1 and E_7 worked on token pruning of transformers (instead of head pruning). They liked our image masks in disclosing the tokens’ semantics but also wanted to see similar saliency maps highlighting the importance of individual patches through ablations. These comments provide promising future directions for us to explore.

2.2.6 Discussion, Limitations, and Future Work

Despite many visual interpretation works for DL, *the unique values of our work* come from the following perspectives. First, our work presents a comprehensive interpretation of ViTs and discloses insightful findings. For example, heads with strong self-attentions are dominantly important. Lower- and higher-layer heads show different local/global attention strengths. Also, we summarize all possible attention patterns between patches. These insights open the hood of ViTs and deepen model designers’ understanding. Second, our interpretation triggers model improvement ideas, e.g., pruning heads with repeating patterns. Thus, improving ViTs with our derived insights would be a direct follow-up work. Lastly, although we focus only on the classification task, we believe our interpretations are transferable to ViT-based detection/generation tasks [101], as those tasks also significantly rely on the multi-head self-attentions of ViTs.

Head-Centric v.s. Image-Centric. We want to emphasize that all our analyses are *head-*

centric, and each head’s behavior is analyzed in one and across all images. Specifically, for *head importance*, we provide each head’s *local* importance on one image and *global* importance over all images (Section 2.2.4.3). For *head attention strength*, we present a head’s attention strengths in one image (Figure 2.3-C2) and its strength distribution over all images (Figure 2.3-C3). For *head attention pattern*, the two-axes/heatmap (Figure 2.3-D3, D4) shows the attention pattern of a head from one image, while the scatterplot in Figure 2.3-D1 lays out the head’s attention pattern over all images. From a different perspective, we believe *image-centric* analysis would also lead to insightful findings, e.g., checking if the heads show similar patterns for images of the same class. We plan to explore this direction in the future.

Performance. To guarantee the exploration interactivity, we have pre-computed some of the visualization data. For example, the head importance metrics are computed offline as they can take hours. The partial pruning in Figure 2.3-B2 is performed online and each computation takes about 0.6 seconds on an Nvidia Titan RTX GPU. The head attention strengths and the t-SNE layout for head attention patterns are both computed offline as they only need to be computed once and directly plugged into our system. In terms of storage, the raw attention weights consume the most space, ranging from 11 GB to 178 GB, depending on the number of heads in the studied ViT. Other data (e.g., images, probabilities, t-SNE results) take about 300 MB in total.

Limitations and Future Work. Our head importance analysis relies on leave-one-out ablations, which do not consider the interaction between heads. In some cases, one head could be important only if another head is pruned. The analysis can be further extended to higher-order interactions, which is our planned future work. Second, our current analysis focuses on the attentions between two consecutive attention layers only. In the future, we would like to explore attention aggregation methods, e.g., [95], to interpret heads’ impact across multiple layers. Lastly, we plan to investigate if the head importance, head attention strengths, and head attention patterns show any class-specific or dataset-specific trends. This will help to diagnose class-related performance issues and validate our findings in more datasets.

2.2.7 Conclusion

In this work, we introduce a visual analytics solution to interpret ViTs. Our interpretation was carried out from three perspectives. First, we answer what heads are more important by introducing multiple head-importance metrics. Second, we explain why a head is important by

disclosing its attention strength distribution across image patches in the 2D spatial context. Third, we adopt an unsupervised learning method to exhaustively summarize the possible attention patterns. Through concrete case studies conducted together with multiple experienced deep learning experts, we verify the efficacy of our visual interpretation solution.

Chapter 3

Visual Analytics and Machine Learning Interpretability for Model Evaluation

Model evaluation is a critical step in the machine learning lifecycle, where the quality, reliability, and robustness of a model are assessed to ensure its suitability for real-world applications. Effective evaluation provides insight into how well a model generalizes to new data, identifies potential biases, and measures resilience to noise or adversarial conditions. However, as modern ML models become increasingly complex, traditional evaluation metrics, such as accuracy or F1 scores, often fall short in capturing nuanced behaviors, inherent biases, and potential failure modes. Furthermore, with domain-specific requirements varying significantly, defining what makes a “good” model can be subjective and highly dependent on specific applications and user expectations. These challenges highlight the need for comprehensive and flexible evaluation methods that go beyond simple performance metrics to address the intricate dynamics within complex models.

VA and ML interpretability techniques together offer powerful solutions to the challenges of model evaluation, providing researchers with deeper insights into model behavior. ML interpretability techniques allow for a detailed examination of how a model generates its outputs, making it possible to pinpoint issues across various criteria. Meanwhile, VA offers an intuitive, efficient, and scalable approach to aggregating and summarizing different model misbehaviors across diverse conditions and data groups. By combining VA with ML interpretability, users can move beyond static performance metrics to interactively explore areas where models may underperform or act inconsistently, fostering a more comprehensive and user-centered evaluation process.

Aiming to provide more flexible, user-centered evaluation perspectives, I integrate VA systems with ML interpretability techniques to develop evaluation solutions for applications such as medical predictions and image classification. For medical prediction models, I focus on generating trustworthiness insights for a series of tree-based ensemble models by adopting SHAP values to interpret feature contributions and monitoring their consistency relative to feature values. I also build an interactive VA system that facilitates detailed comparisons from the perspectives of model performance and trustworthiness. In the case of image classification models, I address their vulnerabilities to adversarial attacks. By analyzing each neuron’s receptive field and its overlap with adversarially vulnerable regions, I identify and explain the most vulnerable neurons, diagnosing model weaknesses. These approaches uniquely combine ML interpretability with VA, offering insights that extend beyond single-metric assessments to deliver more comprehensive and actionable evaluations.

3.1 A Visual Analytics System for Multi-Model Comparison on Clinical Data Predictions

Comprehensive health data plays an important role in the provision of high-quality medical care and decision-making. For example, predictive analysis on medical datasets can help clinicians understand the potential risks of an operation for patients with particular characteristics [102] or identify potential deterioration of health status over various treatment periods [103]. To develop useful predictions, ML methods have been used on medical datasets. For clinical prediction tasks, ML methods must satisfy high accuracy when clinicians rely on the predicted results for their decision-making process [104].

However, solely relying on accuracy is not sufficient. The black-box nature of ML methods creates a concern when it comes to critical medical decision-making. For instance, Caruana et al. [13] trained a neural network (NN) to identify pneumonia patients who should be admitted to hospitals as opposed to being treated in outpatient clinics based on their risk profile. Even though this NN model had a high accuracy value, the model determined that pneumonia patients with asthma should not be admitted, thinking that these patients have a lower risk of dying. This dubious prediction is caused by the fact that these severe patients had been aggressively treated

in intensive care units and, as a result, they survived at a high rate [13, 14]. As shown in this example, the decision relied on an ML model might cause critical harm to patients. There is little tolerance for errors in medical diagnosis and decision-making; thus, providing transparency and interpretability in ML models, is critical in validating and reasoning the resulting predictions.

To address this problem, data scientists have developed several methods to provide interpretable information of ML models [14]. For example, these methods can show which features of a patient’s record have a strong influence on the decision from an ML model (called local feature contributions [105]). This information is useful for both clinicians and researchers to judge whether they can trust the model’s decision, using their medical knowledge. However, when they want to select the best ML model from various candidates [106, 107], checking each model’s interpretable information on all patients’ records is not realistic. Therefore, clinicians and researchers need analysis methods that facilitate a comprehensive comparison of the interpretable results across various ML models and different patients.

Although ML models might have similar performances (e.g., high accuracy) in predictions, the models may vary in terms of their algorithms or the insights learned from the dataset. To understand such variation, the first step is to obtain an overview and comparison of the models’ inner prediction rationales. Then, to evaluate the reliability of each model, the focus should be on the consistency of these prediction rationales when applied to a group of similar patients.

To address the aforementioned needs, we develop a visual analytics system for multi-model comparison based on the interpretable information. We focus on analyzing tree-based ML models (e.g., random forest [108] and gradient boosting [109]) because these models are widely used in predictions of clinical datasets [110, 111, 112]. We use model-agnostic interpretation methods [14] for systematic comparisons, in conjunction with several unsupervised methods. Specifically, across the models, we visualize the similarities of local feature contributions for each patient with a dimensionality reduction method. This visualization can provide an overview of how differently the model(s) predicted for each patient’s result. Additionally, our system offers a quantitative method to evaluate each model’s consistency by utilizing “measures of dependence” [113] developed in the statistics domain. By visually comparing the consistency of reasoning across models, users can gain insights into their reliability. Moreover, the information of consistency can help judge whether the users should rely on a predicted result based on which features or which range of feature values contribute to the model’s prediction.

We demonstrate the usefulness of our system by comparing six models tuned with different tree-based ML methods. Through the demonstration, we discuss the feasibility of applying visual analytics upon interpretable results to aid clinicians’ medical diagnosis and decision-making.

3.1.1 Related Work

We survey the relevant works in ML-based prediction on clinical data, interpretation methods for ML models, visualizations for ML interpretability, and consistency measurements of prediction models.

3.1.1.1 Machine Learning-Based Prediction on Clinical Data

For medical predictions on clinical data, two major categories of ML methods are frequently adopted: tree and recurrent neural network (RNN) based methods.

Tree-based methods are appropriate for medical prediction tasks since they characterize decision-making after encountering different attribute values for each patient. Besides the nature of prediction tasks, many clinical datasets are in the form of temporal sequences. In this sense, the tree-based methods are used to forecast a result at a certain future time step based on the results in past time steps. Moreover, extensive research has been conducted to ensemble multiple decision trees. Random forest [108] uses an ensemble of trees to reduce overfitting and improve generalization by averaging over the ensemble. Meanwhile, gradient boosting [109] builds an ensemble of trees sequentially, where each tree corrects the errors of the previous ones to minimize the loss. Further, a series of variations [114, 115, 116] in the implementation of gradient boosting are developed to provide more efficient computations and better performance.

RNN-based methods are designed for temporal classification and can thus be used to make predictions on clinical data. For example, Lipton et al. [117] used an RNN with long short-term memory to classify diagnoses of patients’ temporal sequences. Che et al. [118] further enhanced RNN to address a problem of data with missing values. While deep learning methods are able to provide high prediction performance, these methods typically need a large dataset (e.g., more than tens of thousand records) for their training.

3.1.1.2 Interpretation Methods for Trees and Deep Learning Models

Various interpretable ML and post-hoc analysis methods have been developed. Adadi et al. [14] provided a comprehensive survey of interpretation methods. Here, we describe only the methods related to trees or deep learning based models.

For tree-based models, interpretation methods can be categorized as a **global** scale, which tries to investigate each feature’s impact on all instances’ predictions (global feature contributions), and a **local** or **instance** scale, which describes the contribution of each feature for each instance’s prediction (local feature contributions). There are several ways to calculate feature contributions. For example, Palczewska et al. [105] introduced a method calculating the local increment value from a parent to a child node corresponding to one feature. Another example is the TreeInterpreter [119], which obtains each feature’s contributions by tracing each tree branch from a leaf node to the root.

As for deep learning used for medical predictions, Choi et al. [120] built a variation of RNN, called RETAIN. By integrating the attention mechanism in RNN, RETAIN provides interpretable results while keeping prediction performance of RNN. Also, Kwon et al. [121] modified RETAIN so that it also provides the attention value for each feature for each time step.

Besides interpretation methods for specific models, there are also model-agnostic methods. Model-agnostic methods perform generic model explanations with common components of different models. LIME [17] is an example of an interpreter for any type of model. Another example is SHAP [18], which presents the impact of each feature value on the prediction. For a more comprehensive list of model-agnostic methods, refer to the survey by Adadi et al. [14].

3.1.1.3 Consistency Measures of Model Interpretations

On the most generic level, a machine learning algorithm’s consistency can be defined as its stability when there are small perturbations of the input data. A learning algorithm’s stability was first investigated by Devroye and Wagner [122], in which they observed the quantitative results of leave-one-out error. Since then, many works have been conducted on probabilistic analysis of learning algorithms’ stability [123, 124, 125]. Then, Bousquet and Elisseeff [126] defined the notion of uniform hypothesis stability to derive the generalization error bounds.

In addition to the probabilistic analyses on stability, statisticians also developed the definition and theories of learning algorithms’ consistency. With the definition by Kearns and Ron [127], an algorithm is consistent if it always returns a result that is consistent with the given examples. Then, the definition of probably approximately correct (PAC) learning algorithm [128] is introduced, which is an extension of consistent algorithms.

In our visual analytics scenario, the consistency of each ML method’s inner rationales is of

our interest. Therefore, besides the stability or consistency of the learning algorithm, we also survey the measures of dependency between two random variables, which characterize the dependence of an ML model’s rationales (e.g., local feature contributions) on the actual feature values. To begin with, Pearson’s correlation coefficient provides a measure of the linear relationship between two variables [129]. Later on, Spearman [130] extended Pearson’s correlation to nonlinear relationships. However, the Spearman’s correlation coefficient is limited to monotonic dependencies between variables. With the development of information theory, there are works summarizing the shared information between the two random variables. For example, mutual information [131], maximal information coefficient [132], and total information coefficient [113] have been introduced. All these measures do not have assumptions of the random variables’ distributions and, thus, these measures are more suitable when the distributions are unknown and the relationships are nonlinear.

3.1.2 Analysis Questions (AQs)

As mentioned in the previous sections, the general goal of this study is to *leverage visual analytics when reasoning and comparing multiple models’ interpretations*. Here, we list more detailed analysis questions that we want to answer with our visual analytics system. These questions have led us to design the methods described in Section 3.1.3.

- **AQ1:** Do multiple models (even trained with similar ML methods) have different internal criteria for predictions and how different are they?

An answer to this question will show the importance of interpretations and comparisons of multiple models. For example, the gradient boosting method and its variations have the same theoretical background. We want to first know whether these methods have significant differences in their prediction criteria. Afterwards, we can move on to the next level of analysis.

- **AQ2:** Which model likely has a higher consistency in its prediction criteria and should be more trusted consequently? Furthermore, which range of different features are more reliable within a model’s prediction criteria?

After getting an overview of different models’ consistency, users would like to know the details of each model’s consistency for each feature, so that they can further select the reliable features to trust when viewing the predictions. Moreover, even within one feature, the consistency could vary across each range of feature values. For example, a model could have high consistency in the prediction for patients who are over 60 while having low consistency for younger patients.

Therefore, our system should also support these analyses.

3.1.3 Methodology

We describe our dataset, prediction task, ML methods, interpretation measures, and consistency measures of interpretations. These are used in our visual analytics system, as described in Section 3.1.4.

3.1.3.1 Data and a Prediction Task

We use the MIMIC-III dataset [133], which is a large, open-access clinical database composed of de-identified critical care unit admission records for over 40,000 patients. The dataset includes demographics, vital sign measurements, admission information, test results, medications, procedures, and mortality.

From this dataset, our prediction task is to foresee the chance of in-hospital mortality, given a patient’s current and previous admission records. This large database consists of patients with more than 14,000 types of diagnoses, and thus it is irrational to predict the status of patients with drastically different diagnoses. To concretize our prediction task, and to make the ML models’ interpretable information more reasonable, we extract admission records that contain the same diagnosis, specifically patients diagnosed with atrial fibrillation (AF). Then, we process the database into a tabular dataset, containing extracted relevant features. As a result, we obtain 8 features from 12,886 AF patients. These features include demographic information, admission status, and information within their inpatient stay (e.g., the number of ICU stays, `icustays_num`).

Although we use a specific dataset for the development of our visualization interface, our analysis methods and visualizations are designed to be applicable to other clinical datasets.

3.1.3.2 Machine Learning Methods

For our analysis, we use six different tree-based methods: decision tree (DT) [15], random forest (RF) [108], gradient boosting decision tree (GBDT) [109], light gradient boosting machine (LightGB) [115], CatBoost [116], and XGBoost [114]. These six methods are chosen for their wide usage for clinical predictions and often provide satisfying performance.

As described in Section 3.1.3.3, we use model-agnostic interpretation methods to understand the models’ rationales. Therefore, although we use these six methods throughout the rest of the content of the chapter, our methodology in the ensuing subsections is generic enough to apply

on different ML methods, including deep learning models.

3.1.3.3 Analysis Methods for Understanding Models' Internal Criteria for Their Predictions

We measure feature contributions to compare multiple ML models' internal criteria for their predictions (**AQ1**). Given an ML model and the classes of the prediction target, a feature contribution represents how strongly each feature affects the prediction results. Typically, for a binary prediction task, a feature contribution can be either a positive (contribute to positive class), zero (neutral), or negative (contribute to negative class) value. In terms of granularity, there are global and local feature contributions. A global feature contribution represents the general effect of that feature to the overall prediction across all records whereas a local feature contribution shows an impact of each individual record of a feature to the corresponding prediction.

To answer the first part of **AQ1**—having an overview of multiple models' inner criteria—adopting either global or local feature contributions should be sufficient enough. However, for the second part of **AQ1** and **AQ2**, we should offer comparisons at a more detailed level. Therefore, instead of obtaining an overview of each feature's impact on the predictions (i.e., global contributions), we have decided to measure local feature contributions.

To obtain local feature contributions, for DT and RF, we use the method described in [105], and for the other methods, we adopt the SHAP value [18]. Between the two methods, the SHAP value is model-agnostic, and, thus, can be adapted to measure feature contributions of any other models.

Though we use tree-based ML models for this study's experiments, we still employ model-agnostic interpretation methods for two reasons. Firstly, while the theoretical backgrounds of the tree-based models are similar, each of them still employs a different technique and provides a different interpretation method. Using a model-agnostic method provides a fair comparison across the models. Second, employing such interpretation methods can help our methodology be more generalizable for other potential ML models.

Let $\mathbf{v}_i$ be a vector of feature values of the i -th data record ($i = 1, \dots, n$). $\mathbf{v}_i$ can be represented as $\mathbf{v}_i = (v_{i,1}, \dots, v_{i,j}, \dots, v_{i,m})$ where m is the number of features and $v_{i,j}$ is the j -th feature value of the i -th data record. We obtain the local feature contributions of all features for each data record. Specifically, $\phi_i = (\phi_{i,1}, \dots, \phi_{i,j}, \dots, \phi_{i,m})$ where ϕ_i is the local feature contributions of

$\mathbf{v}_i$ and $\phi_{i,j}$ is the j -th feature’s contribution of the i -th data record. Because there are k models, for each model, we compute a set of such local feature contributions with m features for each of n data records. As a result, in total, we have $n \times k$ feature contributions vectors with length m . For example, our dataset described in Section 3.1.3.1 has $n = 12,886$ and $m = 8$. Also, we compare $k = 6$ models described in Section 3.1.3.2. Thus, in our case, we will get $12,886 \times 6 = 77,316$ vectors with length 8.

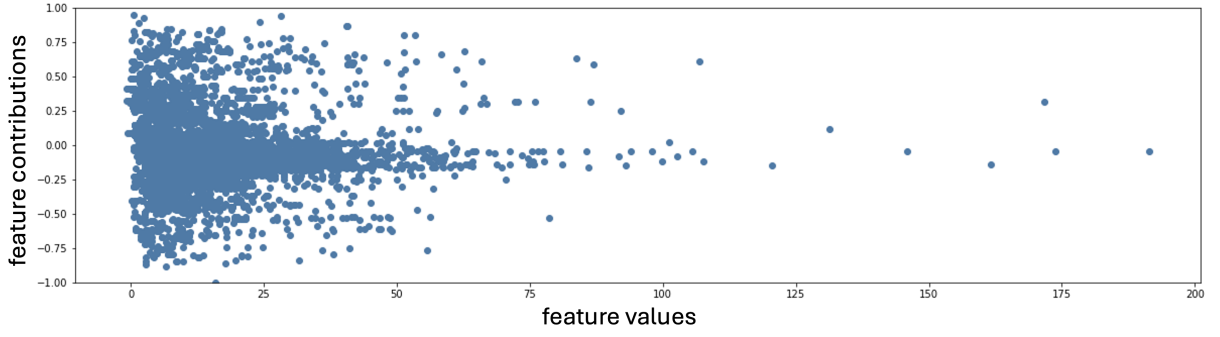
However, it is difficult to review a large set of feature contributions (e.g., 77,316 vectors) one-by-one. Thus, to effectively obtain an answer for **AQ1**, we provide an intuitive overview of the feature contributions’ similarities across different models. The visualized example can be found in Figure 3.5. We adopt DR methods such as t-SNE [62], to project these vectors of dimension m onto a 2D plot. By using DR methods, data points with similar feature contributions will be placed close to each other. In addition, the data points can be color-coded by their corresponding prediction models. Therefore, by reviewing the distributions of clusters visually appeared in the DR result in conjunction with the color information of the prediction models, the users can explore how the local feature contributions vary among multiple models. Refer to Section 3.1.5 for the detailed analysis example.

3.1.3.4 Consistency Measures of Models’ Decision Criteria with Interpreted Information

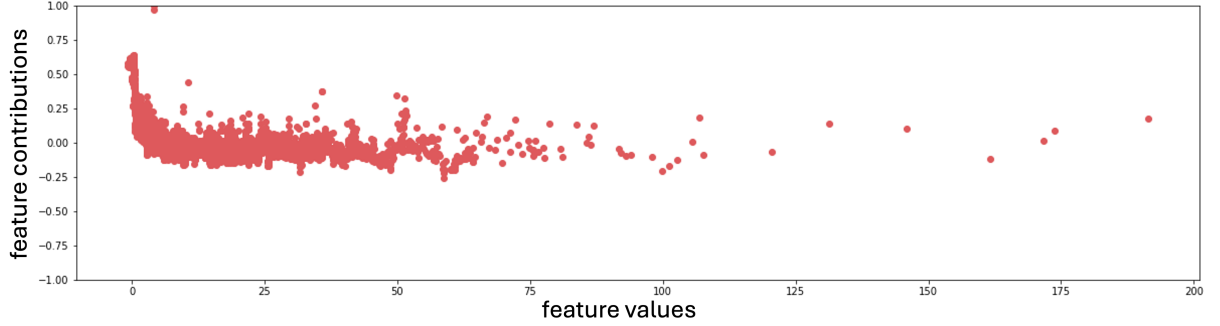
After we compare each model’s internal criteria for its predictions by using the methods described in Section 3.1.3.3, we want to analyze the consistency of each model (**AQ2**). As described in **AQ2**, we consider that a model has high consistency when its criteria for the prediction is robust for a small perturbation in an input feature value. Since local feature contributions characterize how each feature value contributes to the prediction, we concretize the definition of consistency as follows.

Consistency of the model’s internal criteria: The model’s internal criteria have a lower consistency when local feature contributions are more independent from input feature values (the feature contributions are decided randomly regardless of feature values). On the other hand, the criteria have a higher consistency when local feature contributions have a higher dependency on input feature values (the feature contributions are more decisive based on feature values).

For example, two scatterplots in Figure 3.1 visualize the local feature contributions (y -direction)



(a) Model A



(b) Model B

Figure 3.1: The scatterplots of local feature contributions (y -direction) and input feature values (x -direction) for two ML models. Model B’s prediction criteria have a higher consistency than Model A’s.

against the feature values (x -direction) for two different ML models. Here we have the same number of samples for each model (12,886 samples for each). We consider that Model A corresponding to Figure 3.1a has a lower consistency than Model B in Figure 3.1b. For example, for the input values in a range from 0 to 25, Model A has random feature contributions. Thus, for such input values, Model A keeps changing how much it should rely on the corresponding feature. This shows Model A’s low consistency in its criteria for the prediction.

With the definition above, we can obtain consistency with “measures of dependence” [113] between the input feature values and local feature contributions. Measures of dependence capture how strongly two variables are dependent on each other. For example, Pearson’s correlation coefficient is one of the most popular measures of dependence. As we can see the example in Figure 3.1b, feature values and local feature contributions often form non-linear dependency. Therefore, we decide to use measures that can be used to capture both linear and non-linear dependencies. Moreover, it is ideal to use measures that do not have any assumption of the vari-



Figure 3.2: A screenshot of our visual analytics system, which contains the Admission Overview (a), Feature Contributions View (b), Model Summary View (c), and Consistency Charts View (d).

ables' distributions. The recently developed measures, such as the mutual information (MI) [131], maximal information coefficient (MIC_e) [132], and total information coefficient (TIC_e) [113] fulfill the above requirements.

Among these measures, TIC_e is known for the best measure for various datasets [134, 135]. We also tested these three measures on our dataset and TIC_e produced the most reasonable conclusions. Therefore, we have decided to use TIC_e for measuring consistency. Comparisons of these measures are discussed in Section 3.1.6.

3.1.4 Visual Analytics System

We describe our visual analytics system using the methodology described in Section 3.1.3. As shown in Figure 3.2, the system consists of four main views. The left two views, Figure 3.2a and b, are developed for an overall comparison between different models' internal criteria of their predictions (**AQ1**); whereas the right two views, Figure 3.2c and d, can be used for a detailed comparison of the models' consistency (**AQ2**). We provide a demonstration of the user interface as a supplementary video¹.

¹The demonstration of our UI, <https://www.youtube.com/watch?v=KBZYcwEo43Q>

3.1.4.1 Overall Comparison of Models' Interpretations

Using the method described in Section 3.1.3.3, in Figure 3.2a, we visualize an overview of the similarities of each model's local feature contributions, named the Admission Overview as each point represents an admission record with information of the ML model used for the prediction. We employ t-SNE [62] as a DR method because it is suitable to find patterns (e.g., clusters) in a large dataset (77,316 data points in Figure 3.2a). Specifically, we use the openTSNE [136] implementation for the fast computation and precise control of t-SNE's parameters. We color each point based on which model it belongs to. We use categorical colors with enough differences in hues to distinguish from each other. Also, we set color transparency to be able to see overlapped points. By viewing the points' positions for the same model and the distances among points for different models, the user can verify the diversity of different models' rationales. For example, if two models have only a small number of overlaps, they have different prediction mechanisms (e.g., green and cyan points in Figure 3.2a).

The Admission Overview provides a lasso selection with mouse-dragging. The lasso selection allows the user to select a cluster of points. Also, as shown in Figure 3.2a, the user can select multiple clusters. The selected clusters are indicated with the drawn-lasso shapes with the identifying numbers (e.g., ① and ② in Figure 3.2a). Based on the selection, the user can review the detail differences in local feature contributions of the selected cluster(s) from the other points in the Feature Contributions view (Figure 3.2a).

The Feature Contributions View in Figure 3.2b shows a table in which cells contain histograms for the comparison of the distribution of local feature contributions. As shown in Figure 3.2b and Figure 3.3, each row and column correspond to a certain feature and one of selected clusters, respectively. Then, each cell shows the distributions of the local feature contributions of the corresponding feature for the selected cluster and others. As indicated in the legend of Figure 3.3, the pink and gray histograms correspond to the selected cluster and others, respectively, where y -coordinates represent the relative frequency. We have decided to use these two colors to differentiate the categorical colors in the Admission Overview, which are used to represent the models. We should note that since the selected points could be members from multiple different models, we cannot assign the same color used for the model instead of pink. By comparing the height of pink and gray bars, we can understand the differences of the selected cluster's prediction criteria from others. For example, in Figure 3.3, cluster 2 tends to have higher feature

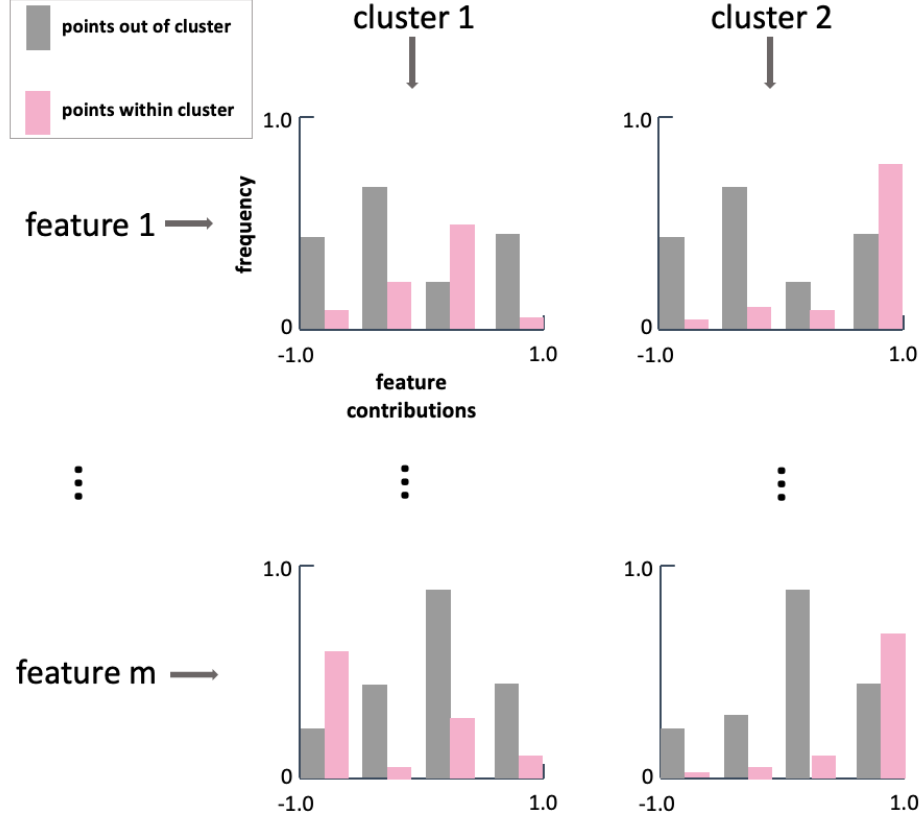


Figure 3.3: An illustration of Feature Contributions View's organization

contributions for feature 1 compared to that of other points. Thus, we can say cluster 2 highly relies on feature 1 for its prediction. Also, by comparing the histograms of each row, the user can observe which feature's distribution varies across multiple selected clusters.

Through the analysis using the Admission Overview and Feature Contribution View, users can understand which features have high contributions to the predictions of multiple selected sets of points. Together with the information of the selected points' models, users can move on to the analysis of the consistency using Figure 3.2c and d.

3.1.4.2 Comparison of Models' Consistencies

After understanding the general differences or similarities among multiple models, we move on to the comparison of different models' consistencies in their inner rationale of decision-making (AQ2).

We first visualize the dependencies between each feature's contributions and the values in the Model Summary View (Figure 3.2c). In the scatterplots of this view, each point's y -coordinate represents the performance measure (accuracy (ACC) or area under the curve (AUC)) of the

corresponding model. The user can select one of these performance measures. Each point's x -coordinate represents a value of the measure of dependency, specifically TIC_e in our case. While ACC or AUC is the measure for each model, TIC_e is the measure for each feature of each model. Thus, we use a horizontal line to represent each model and then, within each horizontal line, each circled dot conveys TIC_e for each feature. Additionally, we use the rectangle shape to indicate the average TIC_e of all features to show the overall consistency of the model. For the model information, we use the same categorical colors with the Admission Overview (Figure 3.2a) to link the two views. Since the measures of prediction performance and consistency are encoded in the x - and y -coordinates respectively, it is intuitive to observe that the models whose rectangular dots are closer to the upper right corner of the plot produce more accurate and trustworthy results. Across the different features for each model, the predictions that highly rely on features (circled dots) that have high TIC_e can be more trusted when the corresponding model's rationales are viewed.

By hovering over each point, the user can see its detailed information (feature name, x , and y values). Also, all circled dots corresponding to the hovered feature will be highlighted with gray outer-rings in horizontal lines of other models for clearer comparisons. Note that black outer-rings show the selected circled dots as explained in the description below. The hovered example can be seen in Figure 3.7. This summary view provides the user with an overview of how different features of different models distribute in terms of their consistencies and the prediction performance.

After obtaining the summary of the consistency of each model and each feature within a model, the Consistency Charts (Figure 3.2d) can be used to verify the results in the Model Summary View as well as compare the consistency of different ranges of feature values. To choose the features and models the user wants to analyze in detail, the user can select one or multiple circled dots which correspond to a certain feature (i.e., points with the same feature name, but different colors).

For the selected point(s), the Consistency Charts (Figure 3.2d) provide a visual explanation of the calculated consistency in the Model Summary View as well as a more detailed inspection on the level of feature values. Because the features we extracted from the MIMIC-III dataset are either continuous or categorical, we provide a different visualization for each type of feature. Users can switch between different features by clicking on the points in the Model Summary

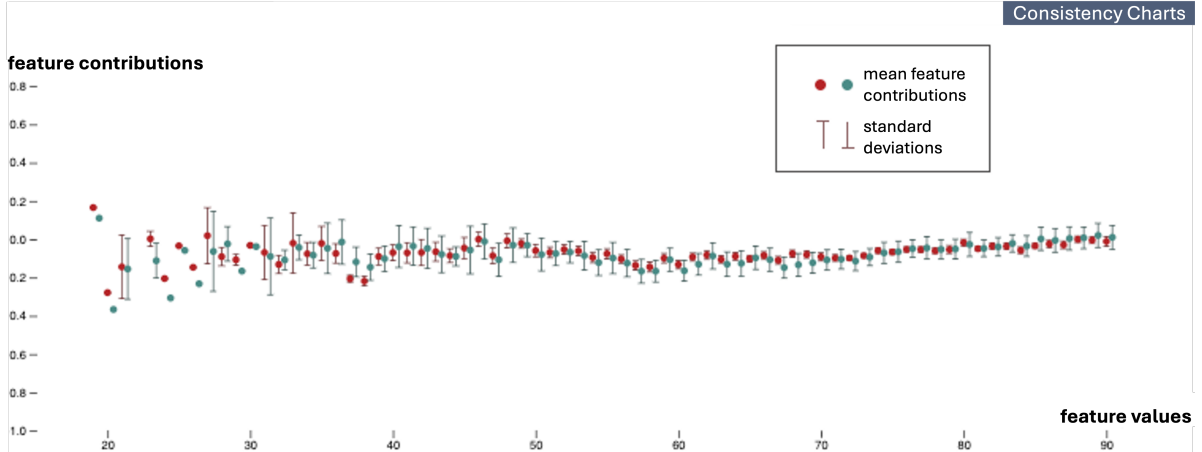


Figure 3.4: The Consistency Charts when visualizing categorical features

View. For continuous features, in the top view of Figure 3.2d, we provide scatterplots of feature contributions (y -coordinates) against feature values (x -coordinates) as similar to the plots in Figure 3.1. To be informed about the selected models, we use the same categorical color with the Admission Overview and the Model Summary View. For each model's scatterplots, we also provide a regression line that shows how the feature contributions change in relation to the values. Since the relationship between feature contributions and feature values is often non-linear, we adopt LOESS regression [137], which is a widely used non-linear regression method. Then, in the bottom view of Figure 3.2d, we also plot the residuals of the regression using y -coordinates. x -coordinates represent feature values as similar to the top view. This plot can assist the user to quantitatively measure how well the feature contributions behave with the distribution of feature values. Comparisons between models can be done by overlapping the scatter plot and the residual plot of each model. For categorical features, we provide a plot of points representing mean values and error bars of the local feature contributions for each feature value, as illustrated in Figure 3.4. To simplify comparisons, we plot the points and error bars of different models with small gaps between their x -coordinates, rather than overlapping them on the same x -coordinates.

By reviewing how widely points are distributed along the y -direction for each x -coordinate, the user can understand which model and/or which range of feature values has higher consistency. For example, in Figure 3.2d, we can see that the yellow model generally has higher consistency than the blue model. Furthermore, within the yellow model, when the feature values are smaller, the residuals tend to have higher absolute values. Thus, when the yellow model predicts the result

Table 3.1: Performance of different tree-based methods for prediction label “in-hospital survival” or “in-hospital death”

Method	AUC	ACC
DT	0.845935282	0.846111720
RF	0.908171950	0.907813070
GBDT	0.902461079	0.901788974
LightGB	0.894114838	0.893391749
CatBoost	0.905595369	0.905074845
XGBoost	0.910045645	0.909638554

for the patients who have low values for this feature, the model’s prediction criteria have low consistency.

3.1.5 Case Study

Using the preprocessed MIMIC-III dataset (refer to Section 3.1.3.1), we compare the six ML models described in Section 3.1.3.2 in terms of their prediction performances, internal rationales for predictions (**AQ1**), and their consistencies (**AQ2**).

3.1.5.1 Models’ Prediction Performances

As described in Section 3.1.3.2, we trained six models with DT, RF, GBDT, LightGB, CatBoost, and XGBoost. We then obtained area under the curve (AUC) and accuracy (ACC) for each model, as shown in Table 3.1. From Table 3.1, in terms of prediction performance, XGBoost has the best performance while other methods excluding DT have similar performance with XGBoost.

3.1.5.2 Overall Comparison of Models’ Internal Prediction Criteria (**AQ1**)

After standardizing different models’ local feature contributions for each patient, we performed t-SNE on these feature contributions. The t-SNE plot is color-encoded and shown in the Admission Overview, as demonstrated in Figure 3.5.

From the overview of similarities of local feature contributions, we can observe a divergence in the positions of the points representing the feature contributions, whereas points that belong to the same models tend to be more clustered together. However, there are exceptions for such cases. For example, the points of models CatBoost (yellow) and GBDT (red) have many overlaps and no distinguishable boundaries, although some areas are denser with points of one model than the other. This implies that these two models seem to share more similar prediction criteria

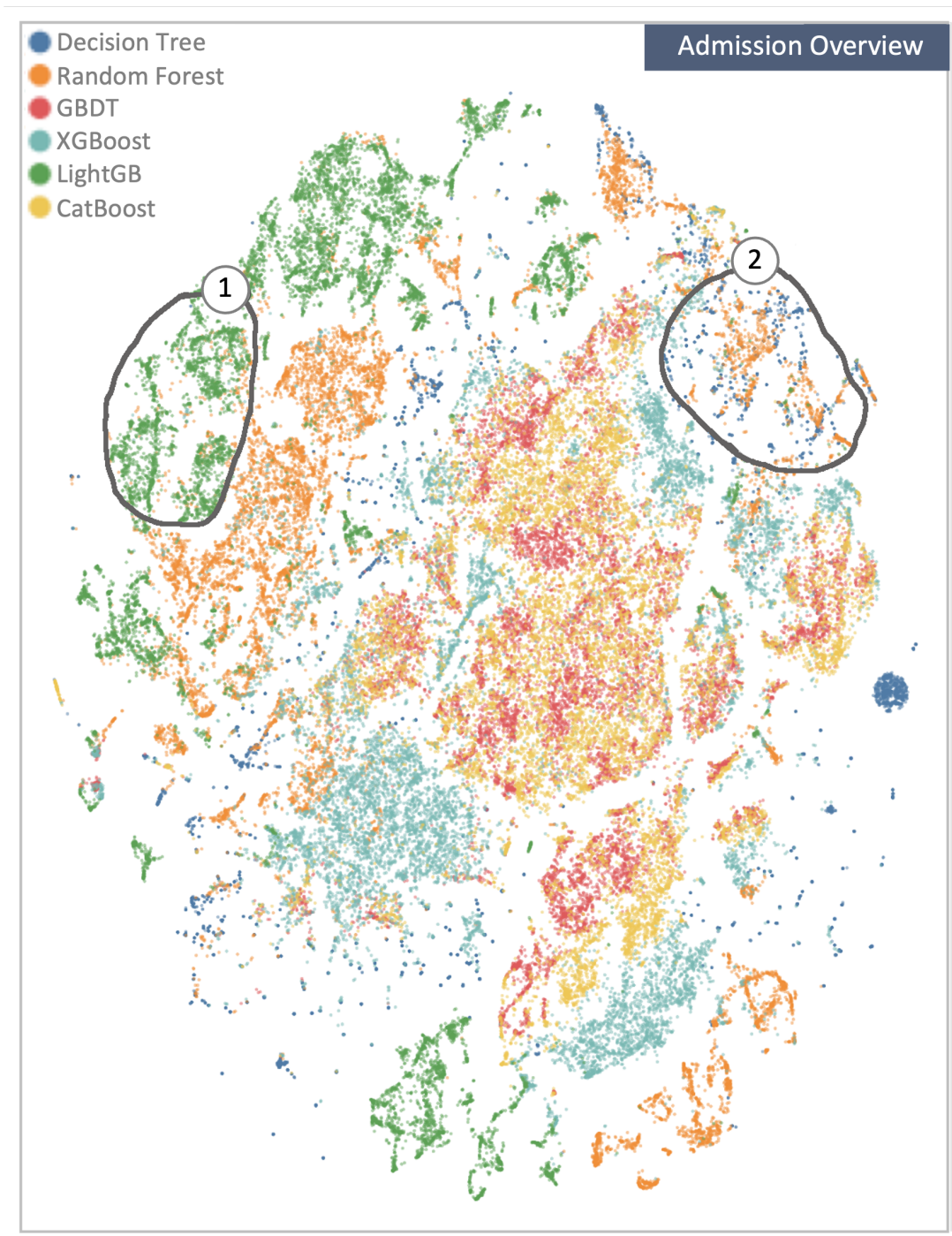


Figure 3.5: t-SNE visualization of model similarities and feature contributions reveals clustering patterns, with CatBoost (yellow) and GBDT (red) overlapping, indicating shared prediction criteria. Distinct clusters appear for other models, such as LightGB (green), which shows intra-model variation based on patient features. Clusters 1 and 2 are lasso-selected to further analyze local feature contributions.

compared to others. However, we can find two general tendencies from Figure 3.5. First, most of these six models' inner criteria tend to have differences from each other as we can see distinct clusters for each model. Second, even within the same model, each model tends to have different prediction criteria based on patients' features. For example, for LightGB (green), while there are several clusters around the top left of Figure 3.5, we can see a distinct cluster around the bottom center.

Then, we selected two clusters of points within the overview to review the differences of their local feature contributions in detail. As seen in Figure 3.5, Cluster 1 contains mostly LightGB points, while Cluster 2 contains mostly DT and RF points. The Feature Contributions View visualizes the local feature contributions' distribution of these two sets of points, as shown in Figure 3.6. By comparing each feature within the same row, we discover that there are significant differences for features `ethnicity`, `age`, `diagnoses_num` (number of diagnoses), `los` (length of stay), and `adm_loc` (admission location), as highlighted in Figure 3.6. In the following subsection, we use feature `los` as an example to demonstrate the use cases of the remaining views.

3.1.5.3 Comparison of Models' Consistencies (AQ2)

We then moved on to the Model Summary view to analyze the differences in consistencies of the feature `los` across models. Here, we focused on the three models of which local feature contributions are included in the selected two clusters (i.e., LightGB in Cluster 1; DT and RF in Cluster 2). As shown in Figure 3.7, the highlighted points correspond to feature `los` of the three models. As observed in Figure 3.7, LightGB's `los` feature has a much higher TIC_e than DT's and RF's. Therefore, it can be inferred that clinicians can rely more on `los` when they adopt LightGB model while they cannot rely as much on `los` when using DT or RF.

By clicking the corresponding points for feature `los` of LightGB and DT as an example, we analyzed the detailed differences of these two models' consistencies with the Consistency Charts, as shown in Figure 3.8. Through the comparisons between the overlaying scatterplots, we can see that LightGB has more consistency than DT because the points tend to be more closely distributed around the regression line. Furthermore, when looking at the residual plots, as shown in Figure 3.8, we observe that DT's residuals tend to have a wider range for small values of `los`, whereas LightGB tends to have small residuals for any values of `los`. Through this view, we can say that LightGB's trends of `los`'s feature contributions can be more trusted.

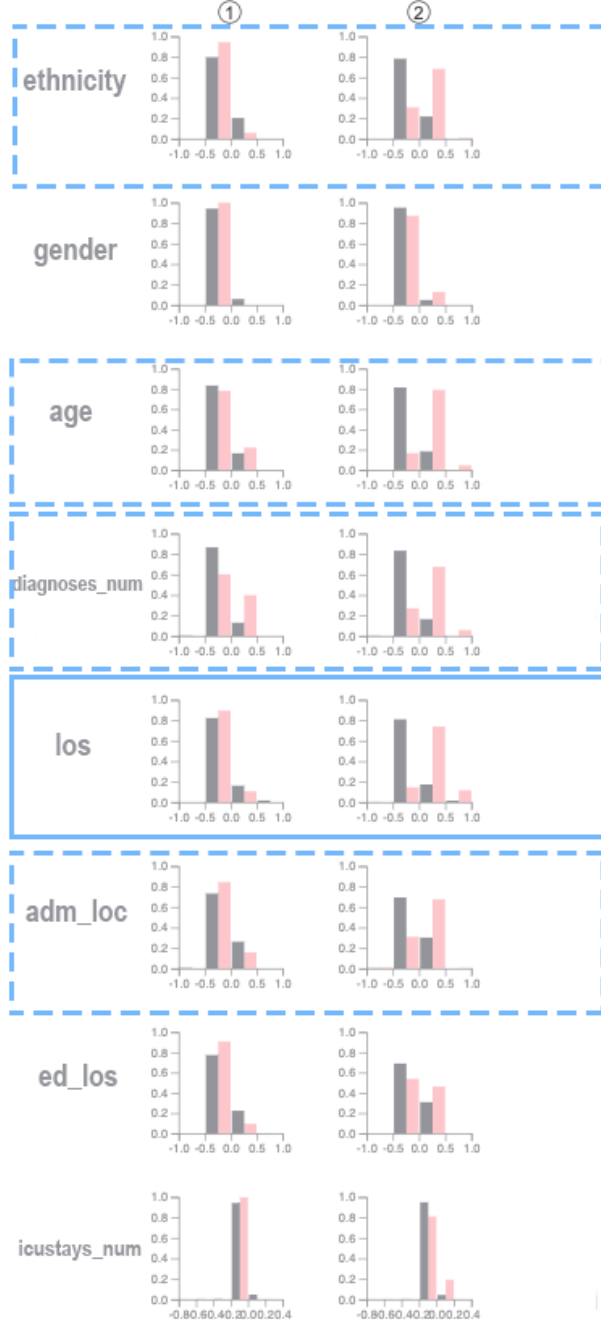


Figure 3.6: Comparison of local feature contributions between Cluster 1 (left column, LightGB points) and Cluster 2 (right column, DT and RF points) using the Feature Contributions View. Differences are observed across features, with significant variations in **ethnicity**, **age**, **diagnoses_num**, **los**, and **adm_loc**, as highlighted. Feature **los** is further examined in subsequent views to demonstrate additional use cases.

Through this case study, we demonstrate how effectively we can answer **AQ1** and **AQ2** with our visual analytics system. Here we have only shown a certain flow of analysis. However, we

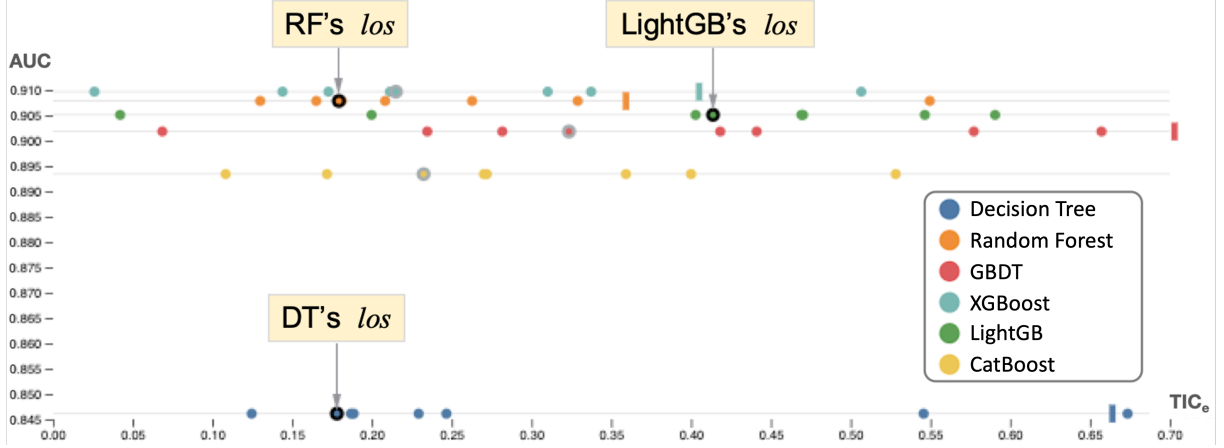


Figure 3.7: Overview of each feature's consistency values for each model

can try various different sets of selections and gain more comprehensive insights. For example, the user would also want to select and analyze different points in Figure 3.5. Such analysis and exploration can be performed with the flexible interactions supported in our system.

3.1.6 Discussion

We discuss our algorithm choice and visualization design; then the limitations of our methods and future work.

3.1.6.1 Dependency Measures

During the selection process of algorithms, we tried several measures of dependency between two random variables to characterize consistency. Since we used the Consistency Charts to verify the calculated dependency values, we also performed comparisons of the dependency measures with this view. As stated in Section 3.1.3.4, MI, MIC_e , and TIC_e can be used to measure the dependency between the feature contributions and feature values. Thus, we tried all these measures and viewed the relationships between the computed dependencies and visual results in the Consistency Charts.

Through this comparison, we observe that although the three measures provided reasonable results for most of the features, MI and MIC_e have capricious behaviors when evaluating dependencies, especially when there are large ranges of feature contributions for the same feature values (i.e., widely different y -coordinates for the same x -coordinates in the Consistency Charts). On the other hand, TIC_e is more stable for any types of features and produces more rational results. For example, as shown in Figure 3.9, both MI (Figure 3.9a) and MIC_e (Figure 3.9b)

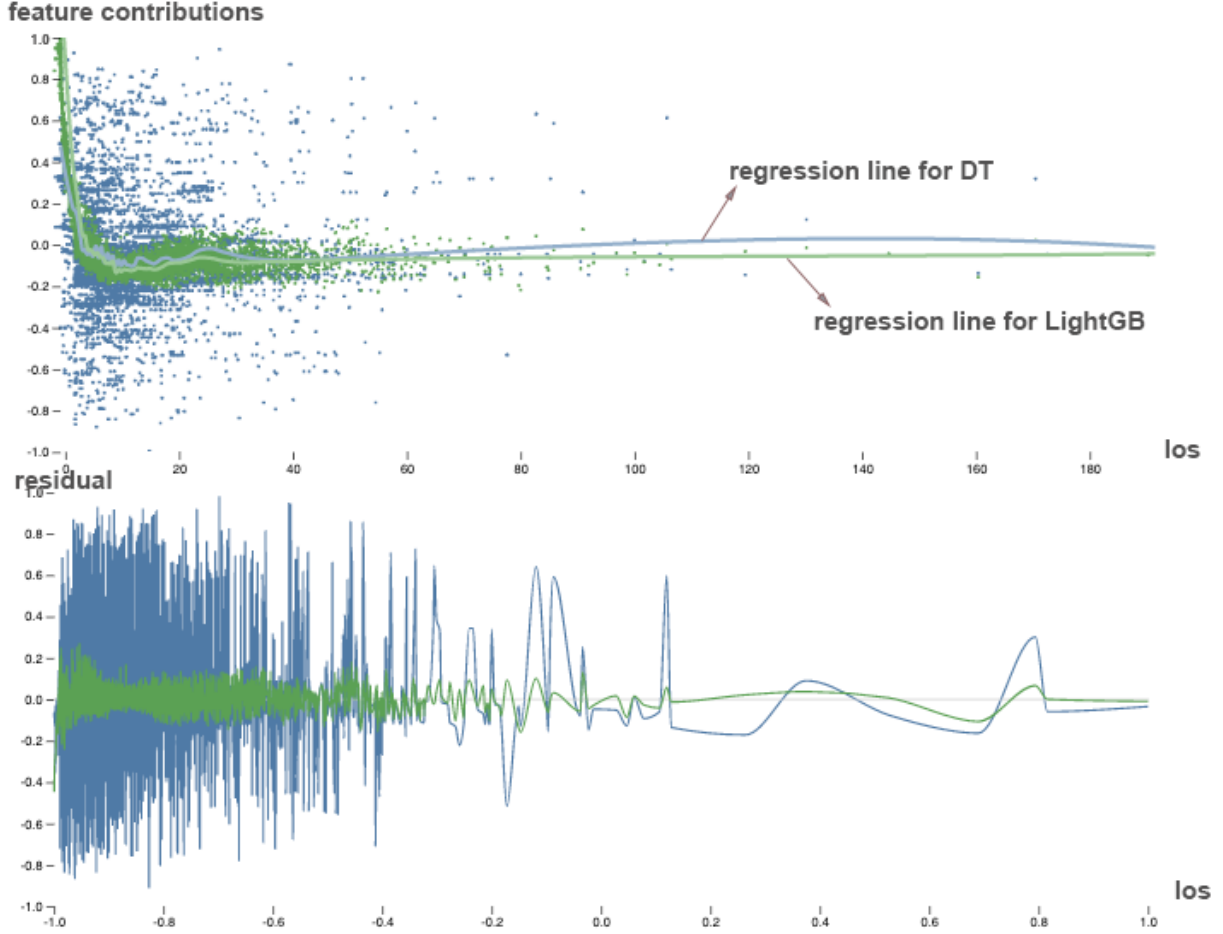


Figure 3.8: Detailed consistency comparison for feature `los` of models DT (blue) and LightGB (green)

indicate that feature `los` of model DT has a high dependency value, even higher than that of CatBoost. However, as shown in Figure 3.10, by reviewing the detailed consistency information, we notice that CatBoost (yellow) should have a higher dependency value than DT (blue). This is because CatBoost’s feature contributions vary less across similar feature values compared to DT. Furthermore, the regression model fitting CatBoost’s feature contributions has much smaller residuals than that of DT, indicating a stronger relationship between feature contributions and feature values in CatBoost. By adopting TIC_e as the dependency measure, the unexpected behavior of the previous two measures is solved, as we show in Figure 3.7. Thus, we eventually chose TIC_e for our consistency measure. This example from our experiment shows that both the analysis algorithm and related visualization can be evaluated by coupling with each other and comparing them; as a result, we can select a better algorithm and/or visualization.

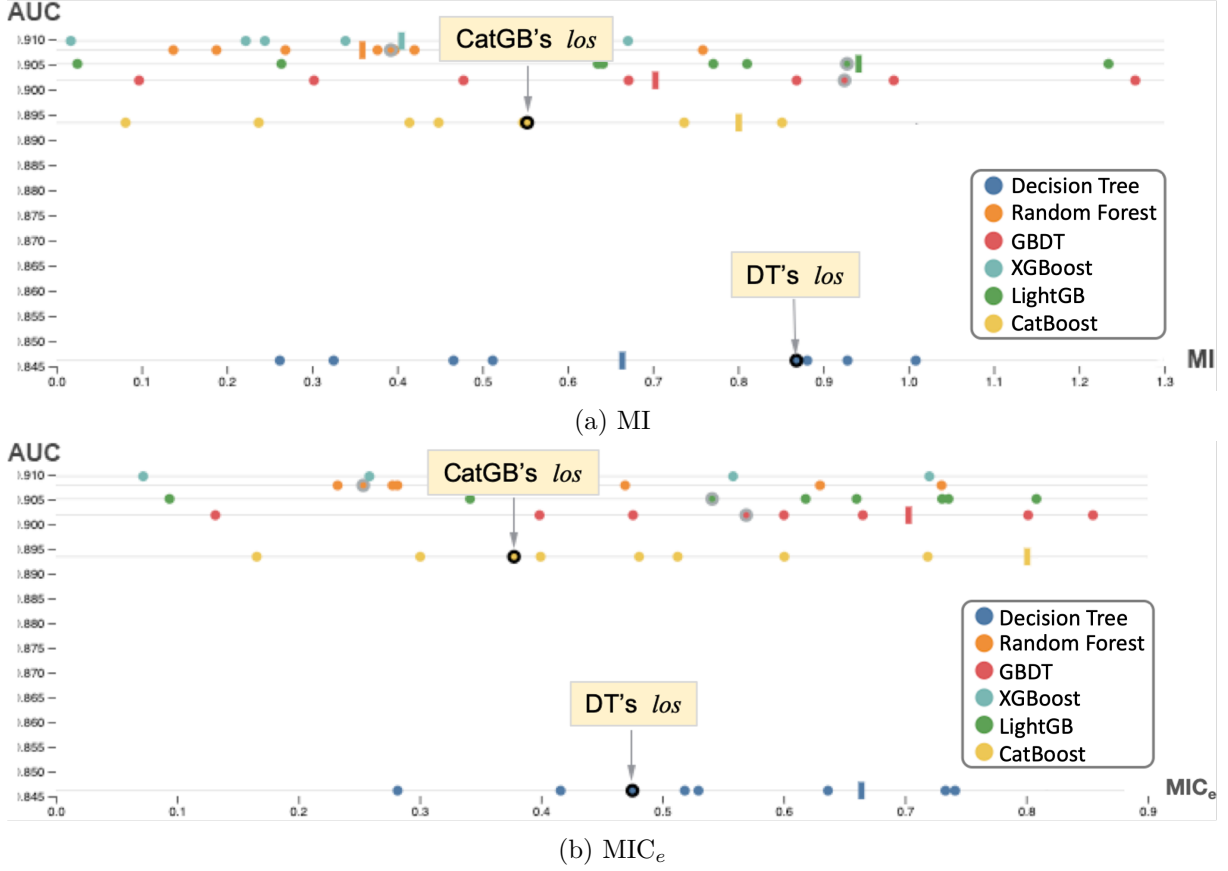


Figure 3.9: The screenshots of the Model Summary View when using (a) MI and (b) MIC_e as dependency measures. In both plots, feature *los* of DT has a higher dependency value than the same feature of CatBoost.

3.1.6.2 Visualizations for Categorical Features

Similar to continuous features, for categorical features, we first designed the Consistency Charts using both scatterplots of distributions (e.g., the top view of Figure 3.10) and an assisting chart that shows how much local feature contributions differ at the same feature value (e.g., the bottom view of Figure 3.10). For categorical features, the counterpart of the regression line of continuous features is the mean or median value for each category; the equivalence of residuals is the error bars calculated around the mean or median (e.g., bars showing standard deviations). However, the mean or median and error bars are usually plotted together. Therefore, instead of showing them in two different views, we first decided to follow this common way (i.e., showing both in one view).

However, as shown in Figure 3.11, by following this format, the visualized results suffered from occlusion and cluttering. This happened because the points (red and teal colored dots)

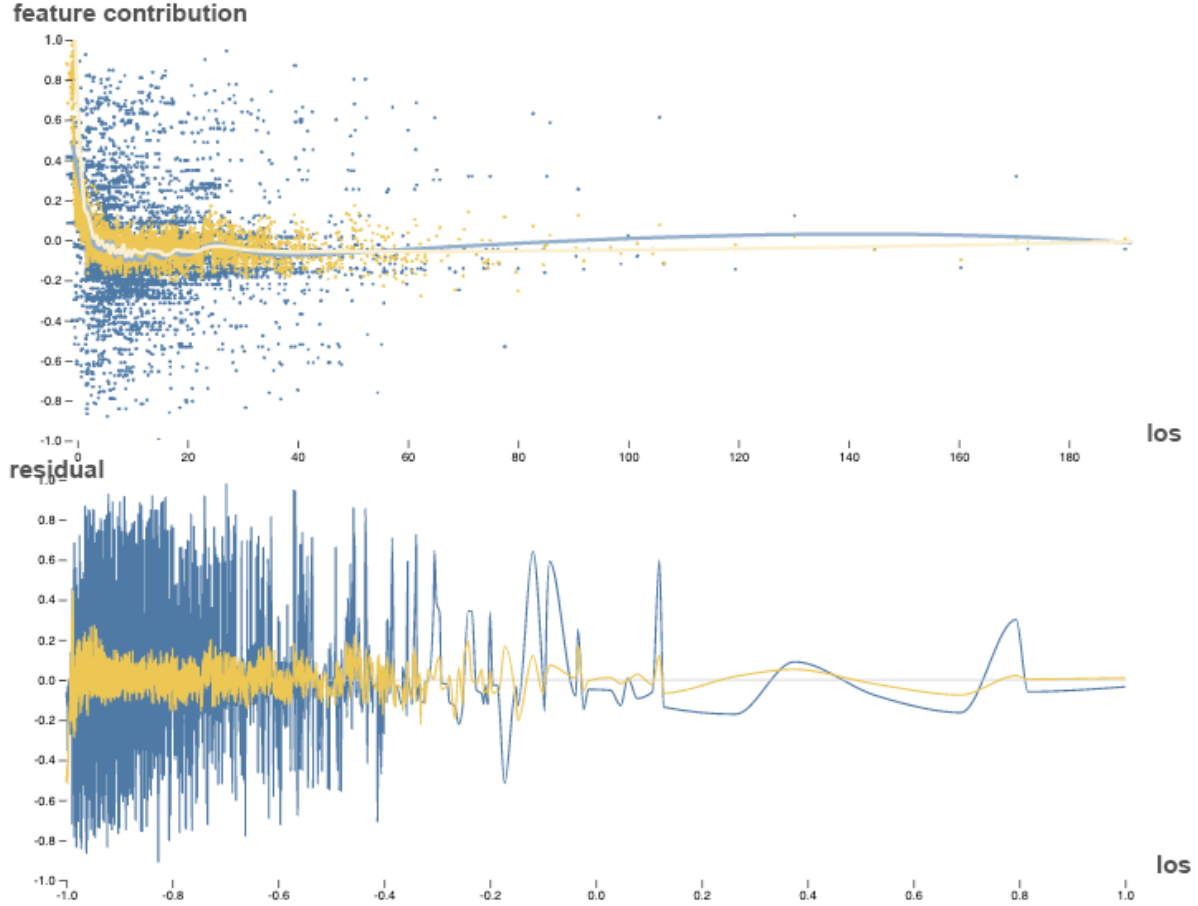


Figure 3.10: The Consistency Charts of LightGB (yellow) and DT (blue).

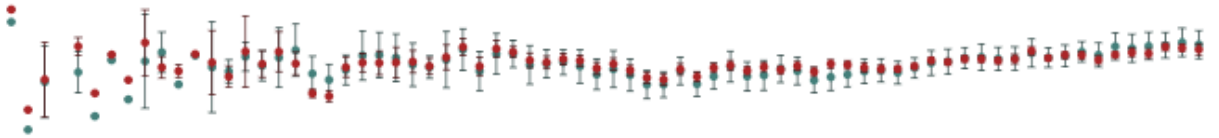


Figure 3.11: Another option we tried for the visualization of categorical features in the Consistency Charts.

and error bars shared the same x -coordinate. Because the categorical values usually would not take many different values (e.g., about 70 in Figure 3.11), we have enough space to use slightly different x -coordinates around each corresponding categorical value. Thus, as shown in Figure 3.4, we tried a plot placing different models' points and error bars with a small gap in x -coordinates. In this way, we were able to view and compare the means and error bars for different models more clearly. Therefore, we decided to use this design for our visualization.

3.1.6.3 Limitations

Scalability for the number of features and ML models. Our visualizations provide enough scalability for the number of data records (e.g., patients). For example, the Admission Overview employs t-SNE [62] for dimensionality reduction and can visualize the overview of similarity of local feature contributions even for tens of thousands of points. However, for the number of features and the number of ML models, our visualizations have limited scalability. The number of ML models we can support is limited because we use colors to indicate the corresponding ML model. Thus, our visualizations can deal with less than about 10 models. We can address this problem by aggregating multiple models based on their similarities in a certain aspect. For example, as shown in Section 3.1.5.2, CatBoost and GBDT have similar distributions of local feature contributions and, thus, the user might want to analyze them as one aggregated model.

For the number of features, we would need to improve several designs if the data were to contain many features. For instance, the Feature Contributions View in Figure 3.2b shows all features’ information by aligning them for each row. This way is reasonable for our dataset consisting of 8 features. However, when there are more than 10 features, showing and analyzing all features’ information is not realistic. In this case, the system should automatically suggest which features the user should review to understand the differences between the selected points from the Admission Overview (Figure 3.2a) and others. For example, we can support such functionality by using the method introduced in [138].

Variety of analyzed ML methods. In the case study of our clinical data predictions, we adopted the tree-based ML methods. In terms of generalization, one future direction is to develop heuristic comparison methods for any kind of ML methods, including deep learning methods. For example, to extend our analysis on time series prediction methods, besides the reliability of each feature, we would also like to compare consistencies of different time steps. For deep learning methods, the reliability of the last few layers’ outputs, for instance, should also be our focus of comparisons.

Supported Analyses. Our methods and visualizations can help understand many aspects of the ML models’ prediction rationales. However, there are still some points that cannot be explained by the current system. For example, while we can analyze how each model relies on some features more than the other features as their learned results, we cannot know how each model obtained such criteria. More specifically, as stated in Section 3.1.5.2, through the

analysis, we found that CatBoost and GBDT tend to have similar local feature contributions. However, we cannot further analyze why these two methods reached such results. Therefore, our work needs to be extended to cover such exceptions. For such analyses, we can incorporate the methods developed for understanding the learning process of ML models. For example, for the tree-based ML methods, we can use the method introduced by Liu et al. [32].

3.1.7 Conclusions

In this work, we develop a visual analytics system that utilizes quantitative methods to observe and compare multiple models’ reliability through their interpretable information. Using our system, insights of multiple models’ internal criteria can be obtained and their reliability can be further evaluated on both overview and individual feature levels. Through the case study, we have demonstrated the usefulness of this visual analytics system to aid clinical researchers’ model selections. Our visual analytics system can be extended to provide additional support for various machine learning methods and a more scalable interface that provides functionalities for a more comprehensive analysis.

3.2 Visual Analytics of Neuron Vulnerability to Adversarial Attacks on Convolutional Neural Networks

Recent advances in convolutional neural networks (CNNs) have achieved remarkable prediction performance in many applications, including medical diagnosis [139, 140], object detection [141], and autonomous driving [142, 143]. However, researchers have found that a subtle but intentional perturbation in a CNN’s input image, called adversarial attacks [144], could cause a malfunction in a high-performance CNN. This phenomenon poses a severe threat to applications less tolerable to safety or security issues, such as road sign recognition of an autonomous driving system [145].

Due to adversarial attacks’ critical impact on practical applications, research efforts have been devoted to understanding how adversarial attacks succeed and to defend CNNs against them [146]. Pertinent to our work, multiple visual analytics works [40, 147, 39] take an approach to identify critical neurons for adversarial attacks by comparing neurons’ activation pathways on original and perturbed images. Another visual analytics approach [148, 40] using image feature

synthesizing methods [149, 150] is to understand each neuron’s role in the prediction through examination of image features (e.g., cat ears) that activate a neuron. However, both of these approaches have limitations. The extraction of the activation pathways is often very complicated, both logically and computationally, limiting their capability of being reproduced or scaled to large CNN architectures. Most of the methods used to understand the neuron’s role can only focus on reviewing a single neuron at a time and also the feature synthesizing methods involve various types of randomness in its feature generation process, consequently, making it hard to compare neurons’ roles and their collective behaviors.

Motivated by the above limitations in the existing works, we introduce an efficient and intuitive method to interpret adversarial attacks from two aspects: (1) *which neurons are more vulnerable to an attack* and (2) *what image features these vulnerable neurons capture during the prediction*. For the first aspect, we apply data perturbations to both input images and neurons and then sort the neurons with multiple metrics that we designed to identify the most vulnerable neurons. For the second, we utilize receptive fields (RFs) [151], which reveal an input image’s area that greatly stimulates each neuron and visually compare the RFs for a set of images before and after the perturbations. The computation of RFs for all neurons requires only a single forward propagation of a CNN, which is computationally much more efficient compared to other image feature visualization methods. In addition to the analysis of a single neuron, to efficiently explore a large number of neurons and interpret their semantics, we support the exploration of a group of neurons clustered in a hierarchy based on their roles in the prediction.

We further integrate these functionalities into a coordinated visual analytics system in order to allow domain experts to understand adversarial attacks through interactive exploration of the two aspects for multiple different input images and neurons. Specifically, our system first facilitates the exploration of the input images and their attacked counterparts (i.e., the *image space*) by providing an overview of the images and sorting them by their perturbation magnitude or prediction probability changes. Second, focusing on an image of interest, our system helps domain experts efficiently explore a large number of neurons from a CNN (i.e., the *neuron space*) by sorting them based on their vulnerability and clustering them based on their role similarity. Lastly, the coordination among multiple views of our system empowers effective analysis across the two spaces and leads to meaningful interpretations of different attacking scenarios. We conduct case studies together with domain experts to validate the findings and derived insights.

In short, the contributions of our work are as follows:

1. We introduce multiple metrics to rank neurons by their vulnerability in an attack.
2. We facilitate the exploration/comparison of the excessive number of neurons in a CNN with a hierarchy that is constructed based on the individual neurons' roles in the predictions.
3. We design an integrated visual analytics system supporting flexible explorations across multiple input images and neurons to help domain experts gain actionable insights from the interpretation of adversarial attacks.

3.2.1 Related Work

Our work addresses the interpretation of adversarial attacks on CNNs, which falls under the research for explainable machine learning (ML). We highlight representative research on the related topics.

3.2.1.1 Visualizations for Explainable Machine Learning

A rich amount of visualization solutions have been introduced recently for the understanding, diagnosis, and improvement of ML models [88]. These visualizations cover traditional ML models (e.g., [38, 152]), ensemble learning models (e.g., [32, 33, 29]), and deep learning models (e.g., [147, 40, 85]). They have successfully proven that visualization can help machine learning models from multiple perspectives, e.g., transparency, reliability, fairness, and robustness [147, 40, 38]. The robustness of deep CNNs (when being attacked by adversarial examples [144, 153]) is the focus of our work.

3.2.1.2 Visual Analytics for Adversarial Attacks

Adversarial attacks [144] are drawing increasing research attention, so as to their interpretations. From the visualization field, we identified three closely related works. *First*, AEVis [147, 39] extracts the data paths of a *benign* image (i.e., non-attacked image) and its *adversarial* counterpart (i.e., attacked image) from a CNN and investigates the diverging and merging pattern of the paths to interpret when and where the attacks happened in the CNN. *Second*, Bluff [40] separates CNN neurons into four groups, i.e., neurons that activated only for the benign class, only for the adversarial class, for both, and for the attacked images only. These four groups of neurons are connected to form a graph, from which, one can infer what original features were kept and what were newly injected during an attack. *Third*, Ma et al. [38] studies the data poison attacks in conventional regression models and reveals the attacking process in four different levels, instance,

model, feature, and local structure, through an integrated visual analytics system. Our work is different from these three works from several perspectives. First, we focus on revealing the vulnerability of individual CNN neurons/filters and disclose how adversarial attacks can fool these neurons to behave differently. Second, in parallel to the above approaches of extracting critical neurons from the prediction paths of benign and adversarial images [147, 39, 40], we propose several computational-friendly (based on RFs) and logically intuitive (based on perturbations) metrics to order the neurons and interactively identify the most vulnerable ones for analysis. Lastly, our work also enables easy explorations of the excessive number of neurons based on the similarity of the roles they have played during predictions, providing a full picture of a CNN model and the overall impact of an attack.

3.2.1.3 CNN Neuron/Filter Interpretation

Multiple feature visualization techniques have been introduced to disclose what image features individual CNN neurons/filters have captured in the field of computer vision. They can be categorized into two groups based on the computation order employed in the techniques. The *forward techniques* use the forward-propagation of CNNs to understand what input pixels (local features) excite a neuron the most. Receptive field (RF) [151] is a canonical example, which we will explain in detail in Section 3.2.2.2. Discrepancy map [154] is another example in this category, which perturbs individual pixels of an input image and then refers to the neurons’ active-level change caused by the perturbations in order to locate the sensitive features (i.e., groups of pixels). The *backward techniques* rely on the backward-propagation of CNNs to synthesize image features that maximally excite a neuron. For example, activation maximization [149] keeps CNNs’ trained parameters untouched but synthesizes the input image so as to maximize a neuron’s activation through gradient ascent. DeepDream [155] extends activation maximization by initiating the synthesized image with an image to be analyzed to preserve the spatial correlations of image features. Feature visualization [150] is also based on activation maximization but enhances it by improving the diversity of the synthesized features. Our work employs a forward technique, specifically, RF extraction [151], to visualize the captured features of individual CNN neurons/filters, because forward techniques are usually computationally efficient and more intuitive to understand the derived feature visualizations.

3.2.2 Background and Motivation

This section provides a general overview of adversarial attacks, CNNs, and the extraction process of the RF of a neuron. Also, we elaborate the necessity of visualizations for the interpretations of adversarial attacks, motivating our work.

3.2.2.1 Adversarial Attacks on CNNs

Adversarial attacks on CNNs [146, 144] add small imperceptible perturbations to an image to fool the CNNs from correctly predicting the image’s class. The process can be formulated as follows. Given an image x and its true label y ($y = \{1, \dots, c\}$ where c is the number of classes), a CNN denoted as a function $f()$ takes x as input and generates a probability distribution across all labels: $f(x) = [p_1, p_2, \dots, p_c]$. If the CNN is well-trained and can correctly predict this instance, then $\text{argmax}_i(p_i) = y$. Adversarial attacks add a subtle perturbation σ to x , resulting in a new image x' , such that $f(x + \sigma) = [p'_1, p'_2, \dots, p'_c]$ and $\text{argmax}_i(p'_i) \neq y$. $\text{argmax}_i(p_i)$ and $\text{argmax}_i(p'_i)$ are called *benign and adversarial labels*, respectively. Similarly, non-attacked images and attacked images are often called *benign and adversarial images*, respectively. The perturbation σ is often very small, such that a benign image x and its adversarial counterpart x' are visually identical.

In general, implementations of adversarial attacks can be categorized into white-box and black-box attacks [146] according to whether the attacking process needs to know the CNNs’ internal details or not. This chapter adopts a white-box approach using projected gradient descent (PGD) [153], which perturbs the input image with a given perturbation budget so that the cross-entropy loss between the predicted probability and its true label is *maximized*. Besides its popularity and efficiency of adversarial image generation, we choose PGD adversarial attacking method because it is proven as the strongest attack that utilizes the first-order information of the network [153]. However, it should be noted that our visual analytics approach is agnostic towards attacking methods, and will promptly adapt to other methods.

Although researchers are well aware of adversarial attacks, there exist only few clear and easily accessible interpretations of the attacks. Many questions regarding why the attacks could fool CNNs remain unanswered, such as: What kind of image features are more vulnerable to an attack? How the roles of neurons have changed by an attack? The answers to these questions need substantial visual evidence, motivating us to introduce novel visual reasoning techniques and develop a visual analytics system for coordinated analyses.

3.2.2.2 CNN and Neurons' Receptive Field

CNNs are the most powerful image classification model thus far. It usually chains a sequence of convolutional layers, with batch normalization (BN) and pooling layers in between, to extract image features. These features are then combined through dense layers to generate the final class probability (see the blue parts in Figure 3.12). The outputs from each convolutional layer are features extracted in multiple channels (or image patches). Each of the outputs is called a feature map. For example, in Figure 3.12, the extracted feature maps from the last convolutional layer are of shape 14×14 and we have 512 image patches, each of which reflects the extracted output from the corresponding convolutional filter (“neuron” and “filter” are used interchangeably in this chapter as with existing works).

The term receptive field (RF) originates from neuroscience, which refers to the area of a body surface where a stimulus could elicit a reflex [156]. Extended to the context of CNNs, RF refers to the regions of the inputs that affect a particular unit the most [157]. Therefore, the semantics reflected by the set of input pixels is often used to interpret the neuron. Two methods of computing and visualizing RFs are implemented and compared in [154]. The first method is the theoretical approximation, which uses the feature map of the corresponding neuron to do image segmentation based on the selected threshold. The second method tries to capture the RFs empirically using estimations of the discrepancy map. Although the second method produces tighter and more exact RFs, the computation is much more expensive as the discrepancy map calculation requires forward passes for the image copies with noise patches at all possible locations.

In our work, we choose the first method because of its efficiency, and it is sufficient to produce

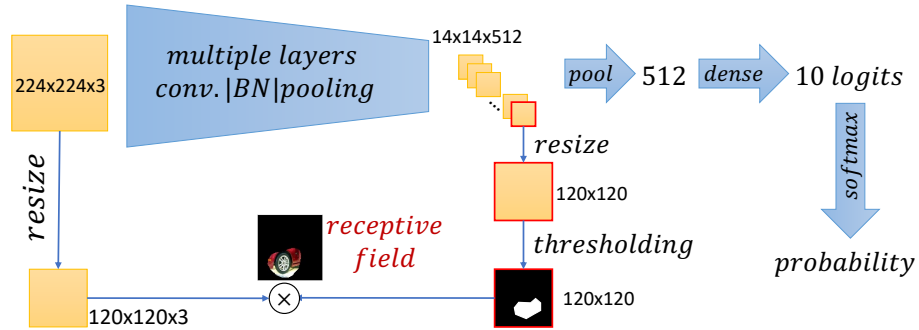


Figure 3.12: A typical CNN architecture (in blue), and the RF computation for a neuron from the last convolutional layer.

reasonable and meaningful results. In detail, we describe this method in four steps as follows, where we use the example in Figure 3.12:

1. Resize the input RGB image to the desired RF size (e.g., from $224 \times 224 \times 3$ to $120 \times 120 \times 3$).
2. For the target neuron at the desired convolutional layer (e.g., last convolutional layer in Figure 3.12), resize its feature map to the desired RF size (e.g., from 14×14 to 120×120).
3. Specify a threshold value to binarize the enlarged feature map from the second step.
4. Apply elementwise multiplication between each of the RGB channels of the resized input and the binarized feature map.

In the second step, the feature map is binarized so that it becomes a mask upon the image in the third step, where only the RF in the image is revealed and highlighted. The threshold value used in the second step controls the number of pixels uncovered. It should be noted that this value is tunable through a slider as described in Section 3.2.4.2. The output from the last step is the RF. For example, the RF result in Figure 3.12 reveals that the target neuron extracts the wheel region of the input image (one image from a `car` class). If we can see the same result across a collection of `car` images, we can conclude that the image feature representing wheels always excites this neuron.

In this work, we focus only on neurons in the last convolutional layer of a CNN to compute the RFs because of the following two reasons. First, these neurons’ RFs are more semantically interpretable than those from the other convolutional layers. As has been crystallized [158, 148], CNN neurons in lower layers (i.e., layers closer to the input) extract basic color and shape primitives (e.g., line or curve patterns), whereas higher layers’ neurons combine previous neurons’ roles to extract object-level features that are more related to semantics (e.g., cat head and car wheel). Focusing on the top level of this learning hierarchy can thus better show the semantics. Second, neurons in the last convolutional layer are right before a dense layer that linearly combines the extracted features to derive the prediction probability. These neurons have a more direct impact on the final prediction probability, which helps us understand the prediction probability changes caused by adversarial attacks.

On the one hand, a convolutional layer could have hundreds of neurons, resulting in hundreds of RFs for a single input image (*one image, many neurons*). On the other hand, the RF analysis of a single neuron also needs the support of numerous input images to extract the shared semantics, and to build confidence during interpretation (*one neuron, many images*). This brings the need

of easy exploration across a space of input images (we call *image space*) and a space of neurons (*neuron space*) to identify images/neurons of interest and interpret adversarial attacks from both aspects.

3.2.3 Requirement Analysis

Based on the stated research questions, the need of visualization solutions discussed in Section 3.2.2, and a review of the closely related literature [147, 39, 40], we extract several common analysis requirements of our target users—ML experts with a demanding need to interpret adversarial attacks. We have distilled and iteratively refined the following requirements for our visual analytics approach and system.

- **R1: Multi-level exploration of the *image space*.** As the number of images generated through adversarial attack experiments usually becomes large and in multiple classes, our first requirement is to explore the image space and locate images of interest. This requires our system to:
 - **R1.1:** provide an overview of the benign images, adversarial images, and their distributions across the predictions;
 - **R1.2:** filter images based on their feature similarity (e.g., similar `cat` images) or prediction similarity (e.g., all `cat` images being mispredicted into `dog`);
 - **R1.3:** prioritize and select images based on attack-related metrics, e.g., perturbation magnitude or prediction changes.
- **R2: Efficient exploration of the *neuron space*.** Even within a single layer, CNNs usually have an excessive number of neurons; consequently, the exploration of the neuron space is nontrivial. To achieve efficient exploration, we should address questions of which neuron should be explored, how we can locate it, and how we can concurrently explore many neurons. Specifically, our system needs to be able to:
 - **R2.1:** effectively identify the most vulnerable neurons to an attacked image and prioritize them for analysis;
 - **R2.2:** intuitively reveal the role of a neuron in the prediction and how its role has been changed by adversarial attacks;
 - **R2.3:** collectively explore neurons that have similar behaviors (i.e., having similar responded image features) before/after adversarial attacks to improve the exploration efficiency.

- **R3: Interpretation of adversarial attacks across the image and neuron spaces.**

Frequently, the image and neuron spaces need to be explored concurrently to interpret adversarial attacks, as the image space carries meaningful semantics and the neuron space encodes the CNN prediction mechanisms. Only when bridging the two spaces, we can externalize how a CNN works through human-understandable semantics. The cross-space analysis could focus on a one-to-one relationship (i.e., how a neuron works on an image), a one-to-many relationship (i.e., how a neuron works on a collection of similar images), or a many-to-one relationship (i.e., how different neurons behave to a focused image).

3.2.4 Visual Analytics Methodology

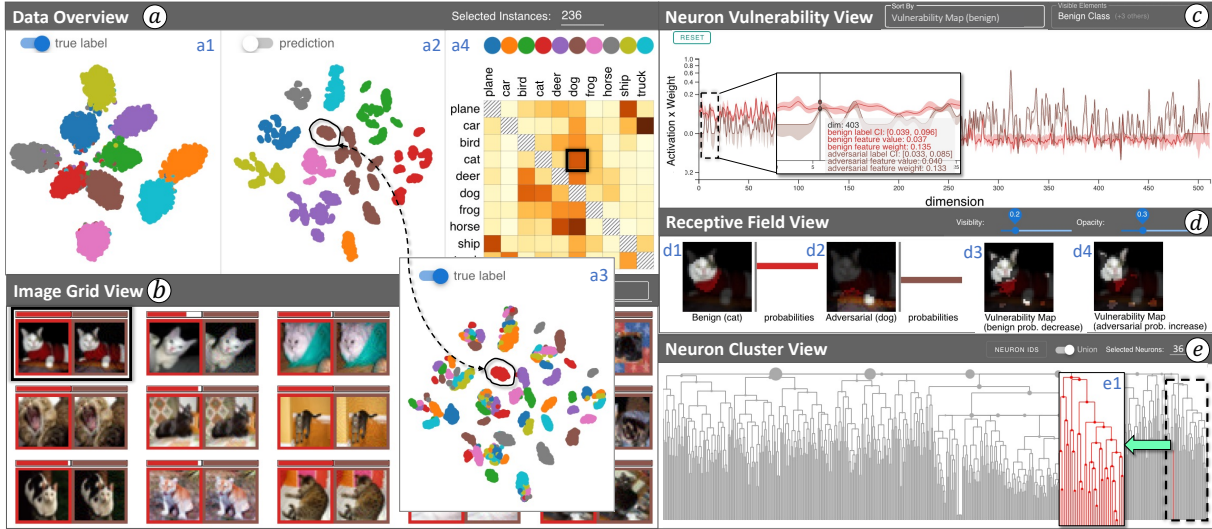


Figure 3.13: Using our visual analytics system to explore and interpret the adversarial attacks. The *Data Overview* (a) visualizes the benign images’ similarities (a1) and the adversarial image’s similarities (a2) based on the CNN penultimate layer’s activation. The color encoding of the points can be switched between using true label or predicted class (a2 and a3). Also, the heatmap (a4) shows the distribution of images based on true labels and adversarial labels. The *Image Grid View* (b) lists benign images and adversarial counterparts corresponding to the selection performed in the *Data Overview* (here the *cat-dog* cell is selected in a4). For the selected image in the *Image Grid View*, the *Neuron Vulnerability View* (c) organizes neurons based on the differences in their values of $\text{activation} \times \text{weight}$. The *Receptive Field View* informs the RFs and vulnerability maps to help interpret each neuron’s behavior. The *Neuron Cluster View* shows the dendrogram constructed based on the similarity of each neuron’s behavior and facilitates efficient interpretation of multiple neurons’ behaviors.

This section describes our visual analytics approach and system. As shown in Figure 3.13, the system consists of five main views. The *Image Space Exploration Component* at the left-hand side of the system provides the *Data Overview* (Figure 3.13-a) and *Image Grid View* (Figure 3.13-b)

to support **R1**. The *Neuron Space Exploration Component* at the right-hand side comprises the *Neuron Vulnerability View*, *Receptive Field View*, and *Neuron Cluster View* (Figure 3.13-c, d, e, respectively) to support **R2**. All five views are coordinated, enabling users to flexibly switch between the data space, neuron space, and connect them for the cross-space explorations (**R3**) to interpret adversarial attacks.

3.2.4.1 Image Space Exploration

Our analysis focuses on test set images that have true positive predictions before adversarial attacks. We use PGD to attack these images, and select the successfully attacked ones only for our analysis. Each of the original images (benign image) also comes with an attacked counterpart (adversarial image). The *Image Space Exploration Component* is designed to explore these images, targeting two goals: (1) provide an overview of the benign images, the adversarial images, and their relations regarding the prediction outcomes from the studied CNN, (2) explore a subset of images of interest (benign-adversarial pairs) and identify significant/representative ones to be further analyzed.

The *Data Overview* (Figure 3.13-a) is designed for the first goal. The two *Scatterplot Views* layout all benign (Figure 3.13-a1) and adversarial (a2) images. The layouts are t-SNE projections using the CNN penultimate layer’s activation (e.g., 512 features produced by a pooling layer in Figure 3.12) of individual images as input. The points are colored by either true labels or predicted labels based on users’ selection. For example, in Figure 3.13-a2 and a3, the prediction and true labels are used, respectively. The *Prediction Matrix View* (Figure 3.13-a4) visualizes a heatmap that presents the distribution of all images, using their true label as row index and adversarial label (i.e., the predicted label of the adversarial image) as column index. For example, if a **cat** image is mispredicted as a **dog** after being adversarially perturbed, it falls into the **cat-dog** cell, which is annotated in Figure 3.13-a4. The darkness of each cell represents the number of instances falling into the cell. Users can select a subset of images to perform detailed analyses with the other views by lasso-selecting of points in either of the two *Scatterplot Views* or clicking a cell of the *Prediction Matrix*.

The *Image Grid View* (Figure 3.13-b) is designed to meet the second goal. The selected images in the *Data Overview* are presented as a grid of image pairs (left: benign image, right: adversarial counterpart). Figure 3.13 shows a result after selecting the **cat-dog** cell from the *Prediction Matrix View*. The border color of each image reflects the predicted label and the

length of the filled color bar above each image shows the predicted probability for the benign and adversarial labels, respectively. To facilitate users to identify the interesting image pairs, this view supports sorting of the image pairs with three metrics: the perturbation magnitude (i.e., the L_2 distance between the benign and adversarial images), the decrease of the benign label’s probability, or the increase of the adversarial label’s probability. As stated in Section 3.2.2.1, the adversarial attack aims to make a mismatch between benign and adversarial labels. This can be caused by decreasing the benign label’s probability, increasing the adversarial label’s probability, or both. Also, the perturbation magnitude represents the general strength of the attack. Thus, the three metrics above are useful to order the images by how intensely/successfully the adversarial attacks disturb the prediction.

With the above views, the *Image Space Exploration Component* allows users to explore the data space in a top-down manner with an “Overview+Details” strategy. The selected image pairs will be sent to the next component for further analysis.

3.2.4.2 Neuron Space Exploration

Focusing on the selected pair of images, we extract the information of neurons in the last convolutional layer from the studied CNN and provide multiple ways to explore them. This section consolidates the exploration and interpretation of a large number of neurons into two questions, and explains our way of answering them.

How to Identify the Most Vulnerable Neurons? A straightforward answer is to sort the neurons based on the difference between the activations of the benign and adversarial images. Although works in some cases, it does not seem to be a robust solution over our explorations due to the intricate connections and the combined effects of different neurons. To systematically answer this question, we develop a more robust sorting by introducing two new perturbation-based metrics that can comprehensively quantify the vulnerability, i.e., perturbing the input images or the output neurons.

Image-Perturbation Metric. Using a single forward-pass of the CNN, we can get the RFs of all neurons (of the last convolutional layer), obtaining a rough idea of which regions each neuron cares about. The next question is which regions are more vulnerable to the adversarial attack. By connecting these more vulnerable regions and neurons that care such regions, we can uncover the vulnerable neurons. We quantify the vulnerability of different image regions by introducing a *vulnerability map*, which is generated as follows (see algorithm 1 and Figure 3.14

Algorithm 1: Generate vulnerability maps by region substitution.

```

1:  $w, h, k, s, c$  // image width, height, region size, stride, number of classes
2:  $bImg, aImg$  // benign image, adversarial image
3:  $bId, aId$  // benign class id, adversarial class id
4: for  $i = 0$ ;  $i < w$ ;  $i += s$  do
5:   for  $j = 0$ ;  $j < h$ ;  $j += s$  do
6:      $[y_1, y_2, \dots, y_c] = CNN(bImg)$ 
7:      $substituteImg = bImg$ 
8:      $substituteImg[i-k:i+k, j-k:j+k] = aImg[i-k:i+k, j-k:j+k]$ 
9:      $[y'_1, y'_2, \dots, y'_c] = CNN(substituteImg)$ 
10:     $bMap[i, j] = y'_{bId} - y_{bId}$  // decrease of benign class probability
11:     $aMap[i, j] = y_{aId} - y'_{aId}$  // increase of adversarial class prob.
12:   end for
13: end for

```

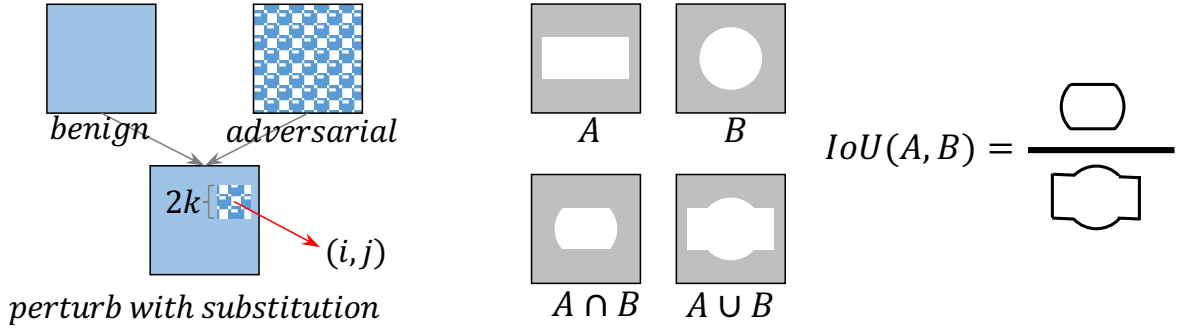


Figure 3.14: (Left) Image substitution/perturbation to probe the image's vulnerability. (Right) Union, intersection, and IoU computation.

left). For a small region of $2k \times 2k$ pixels centered at pixel (i, j) , we substitute this small region of the benign image with the corresponding region of the adversarial image (line 7–8), feed the new image into the CNN (line 9), and record the probability change at pixel (i, j) . We care about both the decrease of the benign class's probability and the increase of the adversarial class's probability (line 10–11). Extending this process to all pixels (line 4–5), we get two vulnerability maps, i.e., the $bMap$ and $aMap$ in line 10–11, sharing the same size with the input. Next, we compute the intersection over union (IoU, Figure 3.14 right) between each neuron's RF and one selected vulnerability map ($bMap$ or $aMap$). Sorting the neurons by their corresponding IoU identifies the most vulnerable ones (in terms of relating to either decrease of benign or increase of adversarial class's probability), as their focused regions have more overlap with the vulnerable regions. In essence, *this metric relies on the perturbation (i.e., region substitution) of*

input images.

Neuron-Perturbation Metric. The feature maps of the last convolutional layer of a CNN are aggregated through a pooling layer to convert them to an n D vector. For example, in Figure 3.12, the 512 feature maps of the size 14×14 are aggregated into 512 values (i.e., $n=512$). Afterward, the dense layer combines the n D vector into a c D vector (c is the number of classes) through a weighted sum to generate the logits for different classes. Note that the logits are values that produce class probabilities after applying normalization with a softmax function. The n D vector, denoting the active level of the n neurons (or called activation), is different for each input image, but should share a roughly similar pattern across images within the same class. We, therefore, compute a confidence interval (CI) of the n D vectors corresponding to images in the same class. As shown in Figure 3.15, we use the blue curve to represent the n D vector of a benign image (each point on the curve corresponds to one neuron's activation), and the blue band to represent the CI of the class that the image belongs to. The same set of information is visualized for an adversarial counterpart using the orange color. The adversarial attack shifts the blue curve to the orange one, i.e., pushing the curve to follow the trend of the orange band. We can substitute/perturb the value of each benign neuron with the adversarial neuron to investigate the final probability change, and use this change to measure the neurons' vulnerability. Sorting with this measured vulnerability is equivalent to sorting the two curves or the two bands by the gap size between them. Our explorations show that sorting by the gap between the two bands provides a more stable result, i.e., class-level neuron activation. The detailed computation of gap sizes (GS) is formulated as:

$$GS((CL_a, CU_a), (CL_b, CU_b)) = \begin{cases} CU_a - CL_b & \text{if } CL_b > CU_a \\ CL_a - CU_b & \text{if } CL_a > CU_b \\ 0 & \text{otherwise} \end{cases}$$

where (CL_a, CU_a) and (CL_b, CU_b) stand for the lower and upper bounds of the CIs for the adversarial and benign classes, respectively. As shown in Figure 3.15 where the above sorting is applied, neurons in region A/C are excited/inhibited by the adversarial attack. Neurons in region B are not significantly affected by the attack. This view is horizontally zoomable to handle the excessive number of neurons.

We consider another fact that even when some neuron has a very high activation, the weight

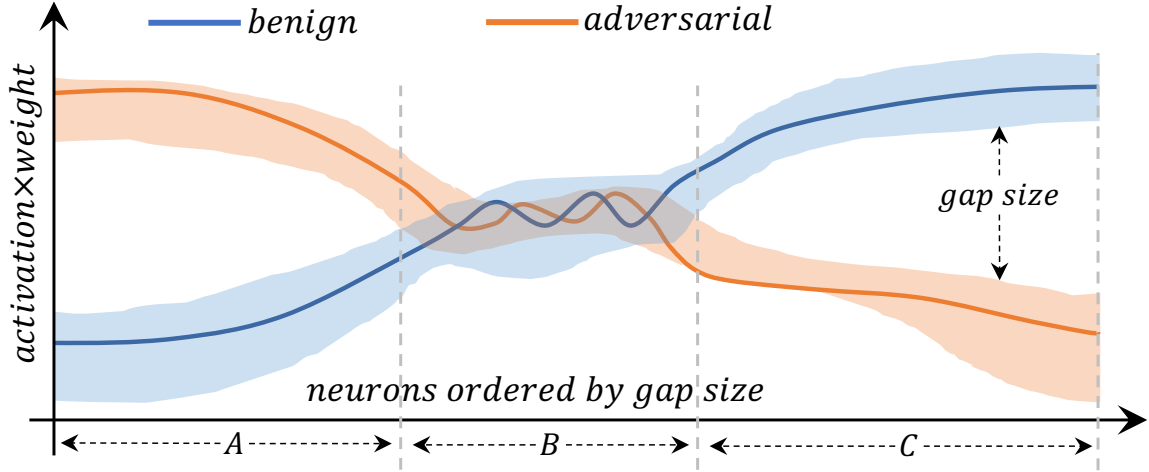


Figure 3.15: Using the gap size to measure the vulnerability (i.e., the shift of the neurons’ activation $\times$ weight) to adversarial attacks.

from the dense layer for the corresponding neuron could be very small. In this case, after the weight and activation are multiplied to produce the logits, the highly activated neuron’s contribution to the prediction could be small. Therefore, to capture the actual impact on the prediction, we use the curve, so as the band, to represent the multiplication result of the n D vector and the corresponding weight from the dense layer (i.e., activation $\times$ weight), instead of the n D vector. Note that the benign and adversarial classes have different weights. For example, Figure 3.12 shows the structure of a 10-class CNN and the dense layer has the weight of a size 512×10 (10 rows). Each row of this weight matrix corresponds to the weight of one class.

Visualization. The *Neuron Vulnerability View* and the *Receptive Field View* are designed to implement the above solutions. The *Neuron Vulnerability View* (Figure 3.13-c) serves as the starting point of the neuron space exploration and shows the plot explained in Figure 3.15. When hovering over any x -position (corresponding to one neuron) over the plot, a pop-up tooltip is shown with detailed information for the neuron. Users can sort the neurons through different metrics introduced earlier (i.e., the image-perturbation metric based on either *bMap* or *aMap* and the neuron-perturbation metric). These metrics help expose the most vulnerable neurons, and users can select one of their interests for further exploration.

The *Receptive Field View* (Figure 3.13-d) presents two types of information to inform the vulnerability of the currently selected image from the *Image Grid View*, together with the semantics of an image feature captured by the neuron selected from the *Neuron Vulnerability View*. The first type of information is the selected neuron’s RFs for the benign image (Figure 3.13-

d1) and the adversarial image (Figure 3.13-d2). The two RFs come along with the predicted probability distribution on the right of each RF. The examples in Figure 3.13-d1 and Figure 3.13-d2 have the probabilities of 1 for **cat** and **dog**, respectively; however, for the case where several classes have a non-zero probability, the view shows multiple bars. The second type of information is the two vulnerability maps: one highlights regions where the image perturbation decreases the benign class’s probability the most (Figure 3.13-d3), the other highlights regions where the image perturbation increases the adversarial class’s probability the most (Figure 3.13-d4). By default, we show the top-20% vulnerable pixels but this can be interactively changed with a slider placed in the view.

How to Explore the Excessive Number of Neurons? Although the neurons can be sorted by the specified vulnerability metric (Section 3.2.4.2), it is still inefficient to inspect, for example, hundreds of neurons one by one. In fact, after users explored the semantics of a neuron, a natural question for the next step is whether or not there are neurons behaving similarly and what they are. If we identify a group of similar neurons, by examining one representative neuron only from the group, we can understand the behaviors of the entire group, making the exploration more efficient.

We address these questions by hierarchically clustering [159] the neurons and visualizing the hierarchy with an interactive dendrogram, as shown in the *Neuron Cluster View* (Figure 3.13-e). As a distance metric used for clustering, we use the L_2 distance between two RFs to measure the corresponding neurons’ similarity (IoU between the two RFs could also be used here). When users select a neuron from the *Neuron Vulnerability View*, its location in the dendrogram is highlighted, as shown in the red line at the far right of Figure 3.13-e. From this location, users can easily identify neurons with similar behavior as similar neurons share the same ancestors and are generally located close to each other in the dendrogram. Also, to see the RFs summarized across similar neurons, the user can click multiple nodes from the dendrogram, which selects the clicked nodes and their descendants. For example, in Figure 3.13-e, the root node of the subtree annotated with the dashed line is selected, and all descendants are highlighted, as shown in Figure 3.13-e1. For all selected neurons, the system computes the union of their RFs, and updates the *Receptive Field View*, accordingly. Users can also switch the *union* operation to *intersection* through a toggle-switch located at the top of the view.

This grouping of neurons based on their RFs’ similarities and hierarchical clustering is a new

approach to providing a summary of neurons’ behaviors. By collectively using the vulnerable-metric-based ordering described in Section 3.2.4.2 and this grouping, we can uncover potentially influential neurons on the prediction that are difficult to identify when only relying on the vulnerable-metric-based ordering. For example, the most vulnerable neuron and neurons that have a behavior similar to it would not be placed closely in the *Neuron Vulnerability View* because of their activation $\times$ weight differences. However, when adversarial attacks aim at the most vulnerable neuron, these similar neurons are also likely influenced by the attacks. Even for the above case, users can first select the most vulnerable neuron from the *Neuron Vulnerability View* and then select similar neurons using the *Neuron Cluster View*. Thus, our grouping approach not only enables efficient neuron space exploration but also supports a more comprehensive exploration of the neuron space.

3.2.4.3 Cross-space Exploration

The image space carries the semantics of different classes, where the mechanisms of how a CNN works on them reside in the neuron space. Domain experts frequently need to jointly analyze both spaces to relate the semantics with the CNN’s working mechanisms, especially when interpreting adversarial attacks (**R3**). We exemplify how our coordinated visual analytics system supports cross-space analysis through three scenarios.

One Image–One Neuron. This scenario explores relations between one selected image and one selected neuron to interpret the neuron’s behavior on a specific image. Specifically, after users select one image in the *Image Grid View* and one neuron in the *Neuron Vulnerability View*, the neuron’s RF is extracted and overlaid onto the selected image. In Figure 3.13-d1, we can clearly see that the neuron captures the face region of the **cat** image. In contrast, the neuron can no longer extract the **cat**’s face from the counterpart adversarial image, revealing the functionality change of the neuron caused by adversarial perturbation.

One Neuron–Multiple Images. The semantics captured by a neuron from a single image may not be able to sufficiently explain the neuron’s behavior. This second exploration scenario allows users to select multiple images, and use the captured semantics from them to gain more comprehensive interpretations of the neuron’s behavior. For example, when clicking the RF of the benign image in the *Receptive Field View*, a pop-up window shows the top- n images that either maximally excite the neuron (sorting option 1), or have the highest prediction confidence (sorting option 2). Each of the images is shown with its RFs to explain what the neuron has captured. On

the other hand, when performing the same interaction on the RF of the adversarial image, the system first extracts the top- n images for the benign counterpart using the same procedure above and then shows a pop-up window listing the extracted benign images’ adversarial counterparts together with RFs. These images work as a context to provide more visual evidence, helping users build their confidence when interpreting the functionality of a neuron. For example, in Figure 3.16 (top-left images), most of the RFs capture the `cat`’s face region, consolidating the behavior of the neuron.

One Image–Multiple Neurons. A large number of neurons raises the need of exploring them concurrently. The *Neuron Cluster View* uses the RFs of neurons on a studied image to cluster the neurons and allows the exploration of RFs in groups. For example, the selection of the dendrogram node in Figure 3.13-e1 extracts all neurons belonging to this tree branch. The union or intersection of their RFs on the benign and adversarial image is visualized in the *Receptive Field View*. When applying the intersection, this exploration reveals the common behavior across the neurons. On the other hand, the union of the RFs shows these neurons’ behavior as a whole.

3.2.5 Case Studies

We first use a small but well-known image dataset, CIFAR10 [160], to demonstrate the usage of our system. The demonstration specifically presents how we answer the two focused research questions using our approach. Next, we use a larger dataset, Places365 [161], to verify the generalizability of our solutions. While we select specific classes in each case for the illustration, attacks between other classes can be investigated and interpreted in a similar manner.

3.2.5.1 CIFAR10 Dataset

CIFAR10 [160] contains 10,000 32x32 RGB images composing 10 classes in the test set. Our studied CNN is a pretrained ResNet18 [162], which correctly predicts 8,200 of the 10,000 images. We then perturb these true positive images with PGD attacks. All images are successfully attacked, and plugged into our system for analysis.

Categorizing Adversarial Images First, we focus on the *Scatterplot View* of adversarial images, colored by the true labels or the adversarial predictions. As shown in the *Data Overview* in Figure 3.13, the adversarial images with the same prediction outcomes tend to form large clusters (Figure 3.13-a2), whereas small clusters of the images with the same true labels exist within each large cluster (Figure 3.13-a3). This pattern indicates that while adversarial images

with the same prediction are perturbed to contain similar features (features of the predicted class), the images with the same true label also have their own attacking patterns.

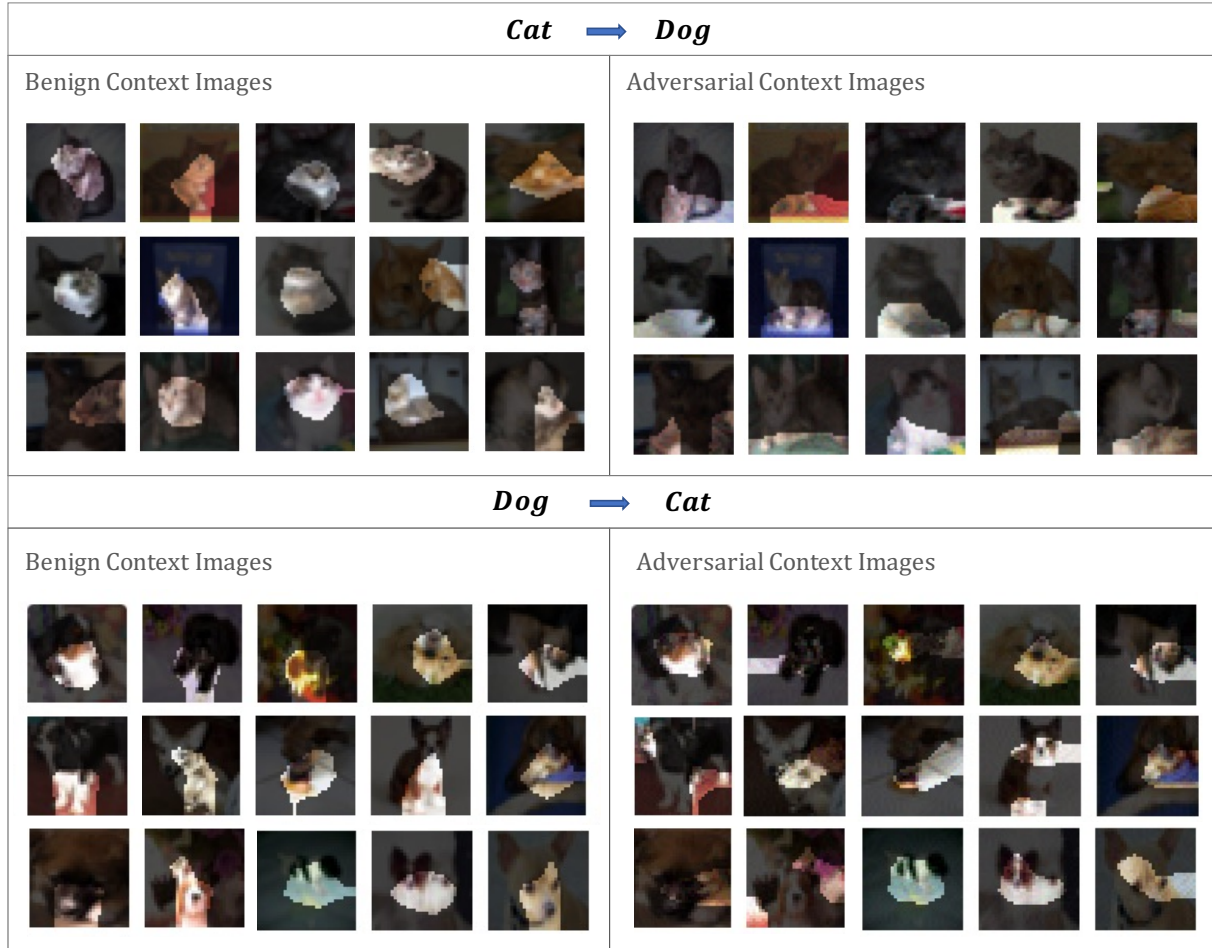


Figure 3.16: Context images for the **cat** v.s. **dog** case. Among the set of **cat** images misclassified as **dog** (top), the cat’s face is captured in most of the benign images (left), whereas the cat’s feet / lower body is captured in most of the adversarial counterparts (right). This explains the different behavior of the same neuron before and after adversarial attack. Furthermore, this explanation can be verified by the RFs of **dog** images misclassified as **cat** (bottom), where dog faces are captured among most adversarial images, but dog feet / lower body are captured among most benign images.

Understanding Attacks with the Image-Perturbation Metric During the above image-space exploration using the *Scatterplot Views*, we notice that there is a large proportion of **cat** images (the red cluster) being perturbed and misclassified into **dog** images (brown), as annotated in Figure 3.13-a2 and a3. We then select this cluster to review their details in the *Image Grid View*.

As shown in Figure 3.13-b, we first sort the images by their perturbation magnitudes; conse-

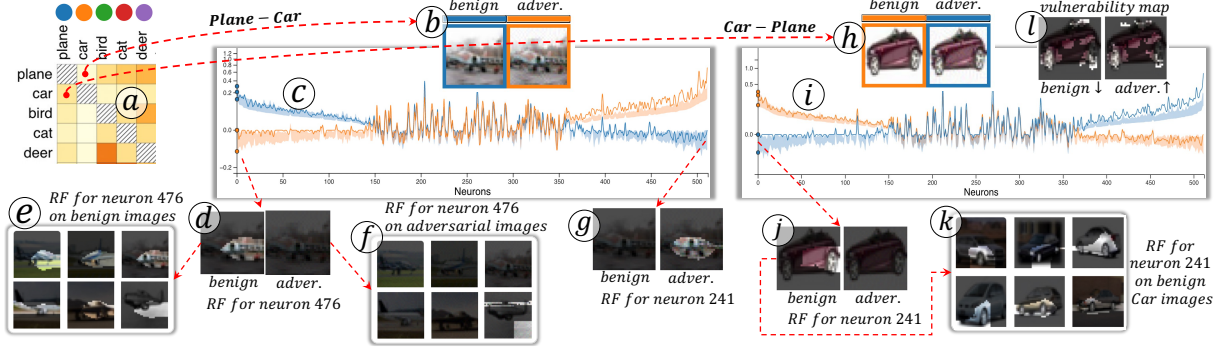


Figure 3.17: The **plane-car** case. The CNN uses neuron 476 and 241 to identify the wheel regions for **plane** and **car** classes respectively (d-e, j-k). The adversarial attack fools the CNN to recognize a **plane** as a **car** by disabling neuron 476 (f) but exciting neuron 241 (g). Consequently, the CNN still extracts wheel features from a **plane** image, but it believes the wheels are from a **car** as they are extracted by neuron 241.

quently, images perturbed more by the attack are arranged at the top of the view. As expected, even for these highly perturbed images, we cannot visually recognize the differences between a benign image and its adversarial counterpart. However, as shown by the bars placed above each image pair, we see that the attack successfully shifts the probabilities. For example, while most of the benign images shown in Figure 3.13-b have $\sim 100\%$ of the probability to be predicted as **cat**, most of their adversarial counterparts have $\sim 100\%$ of the probability to be **dog**. As annotated with the black rectangle in Figure 3.13-b, we select the pair with the largest perturbation for further investigation.

The *Neuron Vulnerability View* is updated based on this selection, as shown in Figure 3.13-c. To reveal which neurons are more vulnerable based on their extracted image features, we sort the neurons by the image-perturbation metric using the vulnerability map (i.e., *bMap* in Algorithm 1). Then, we zoom into the top-25 neurons, as annotated with the dashed line in Figure 3.13-c. Afterward, we select one neuron of interest, neuron 403. We are interested in this neuron because it still has only a small gap in $\text{activation} \times \text{weight}$ between benign and adversarial images and, consequently, the neuron-perturbation metric cannot reveal this type of successfully attacked neurons.

We then examine the corresponding RFs and vulnerability maps in the *Receptive Field View* (Figure 3.13-d). We first observe that the vulnerability maps for both benign probability decrease (Figure 3.13-d3) and adversarial probability increase (d4) contain a large proportion of the **cat**'s face region. This indicates that perturbations around the face region are one reason why this **cat** image gets misclassified as **dog**. In Figure 3.13-d3, we see another focus around the foot region,

which can be considered as another reason for the probability decrease of the `cat` class. We also inspect the RFs of the chosen neuron, as shown in Figure 3.13-d1 and d2. Even though this neuron has similar $\text{activation} \times \text{weight}$ for both of the benign and adversarial images, the RFs are totally different (benign: face, adversarial: feet region). This difference in the RFs provides interpretations of how the adversarial attack alters neuron 403’s behavior (i.e., the focus region changes from face to feet).

To verify this interpretation, we click on each RF to check the context images, which are also `cat` images perturbed to be misclassified as `dog`. As presented in Figure 3.16 (top), neuron 403 indeed tries to capture the face region for benign images but the feet region for adversarial images. This result can be interpreted that the perturbations around the face region disturb neuron 403 to recognize the `cat`’s “face” feature, while the perturbations around the feet region lead neuron 403 to focus on the “feet” feature, which can be easily misunderstood as the `dog`’s. To further validate this interpretation, we select `dog` images that are attacked to be `cat` from the *Prediction Matrix View* and visualize the context images, as shown in Figure 3.16 (bottom). When feeding the benign images of `dog`, neuron 403 captures the feet region, but its RF shifts to the face region for the adversarial images. This provides enough confidence that neuron 403’s role is to capture the face region for `cat` images but the feet region for `dog` images, and the adversarial attack tries to mislead neuron 403 to capture the adversarial label’s feature by weakening the benign label’s feature and enhancing the adversarial label’s feature.

Understanding Attacks with the Neuron-Perturbation Metric Here we analyze the case where the attack causes confusion between `plane` and `car` classes using the neuron-perturbation metric. We first select the `plane-car` cell (Figure 3.17-a) from the *Prediction Matrix View* to retrieve `plane` images (benign label) that are mispredicted into `car` (adversarial label) after adversarial perturbations. In the *Image Grid View*, sorting the images by the change in prediction probabilities, we identify images that the CNN yields very different predictions. One of such image-pairs is shown in Figure 3.17-b. The CNN was nearly 100% sure that this is a `plane` image (the filled blue bar over the left image) but became nearly 100% sure that this is a `car` image after the perturbation (the filled orange bar above the right image). However, it is hard to visually differentiate the benign and adversarial images.

Next, selecting this pair of images, we explore the corresponding neuron space. Sorting the neurons by the “Gap Size” (i.e., using the same sort showed in Figure 3.15), we easily identify

the most vulnerable neurons on the two sides of Figure 3.17-c. For example, neuron 476 is the left-most one, it is active when processing the benign image (blue curve) or images of **plane** (blue band); however, it becomes inactive (close to 0, y -axis value) or negatively active (i.e., decreasing the benign class’s probability) when processing the perturbed image (orange curve) or images of the **car** class (orange band). The change of this neuron decreases the benign class’s probability the most. Checking the details of this neuron on the selected image with the RFs, as shown in Figure 3.17-d, we notice that this neuron extracts the wheel and tail features of the **plane** for the benign image but extracts nothing after the adversarial attack. Clicking the RFs brings a group of representative images and their RFs (6 are shown here), from which, the wheel regions of **planes** are consistently extracted in the benign images (Figure 3.17-e), but nothing is extracted from the adversarial counterparts (Figure 3.17-f). Thus, the adversarial attack seems to successfully stop this neuron from working properly.

Knowing this detail is helpful to interpret the attack but does the adversarial attack prevent the CNN from being aware of the wheel feature? As the next step, we further explore several neurons that increase the adversarial probability the most on the right of Figure 3.17-c, e.g., neuron 241 (the right-most one). This neuron is inactive when processing the benign **plane** image but is active when processing the adversarial image. As shown in Figure 3.17-g, its RF on the selected adversarial image shows the extraction of the wheel region of the **plane**. Similar observations are also found from the context images, confirming that the CNN still extracts the wheel regions after the attack but through a different neuron.

Afterward, clicking the **car-plane** cell (Figure 3.17-a), we switch the analysis to **car** images but mispredicted into **plane** (i.e., the benign and adversarial classes are swapped) and select an image pair with significant probability shift (Figure 3.17-h). When sorting by “Gap Size”, the neurons are sorted based on the class-level aggregated metrics. So the order in Figure 3.17-i is the reversed order of neurons in Figure 3.17-c. The left-most neuron is 241, which contributes the most to the decrease of the benign class’s probability (**car**). As the RF shows in Figure 3.17-j, this neuron extracts the wheel of a **car** on benign images (supported by the context images in Figure 3.17-k as well). However, the perturbed image successfully fools the neuron from extracting this feature. The right-most neuron in Figure 3.17-i is 476, it extracts nothing from **car** images, but can extract the wheel features from perturbed images (not shown in the figure).

Insight. In short, the CNN can extract wheel regions from both the **car** and **plane** class

but through very specific neurons, e.g., in our case, it is 214 for the **car** and 476 for the **plane** classes, respectively. The adversarial attack fools the CNN to recognize a **plane** as a **car** by causing the following situation with the perturbation: (1) the neurons that extract the wheel from **plane** images can extract nothing (2) but the neurons that extract the wheel from **car** images can extract the wheel feature. The CNN then believes the image is a **car** as it saw **car**'s wheels. This also indicates that the wheel feature is a non-robust feature when differentiating **car** and **plane**. During the exploration, we also checked the vulnerability maps for different selected images. For example, Figure 3.17-1 shows those for the **car** image. They verified that the wheel regions of the image are indeed vulnerable to the attacks.

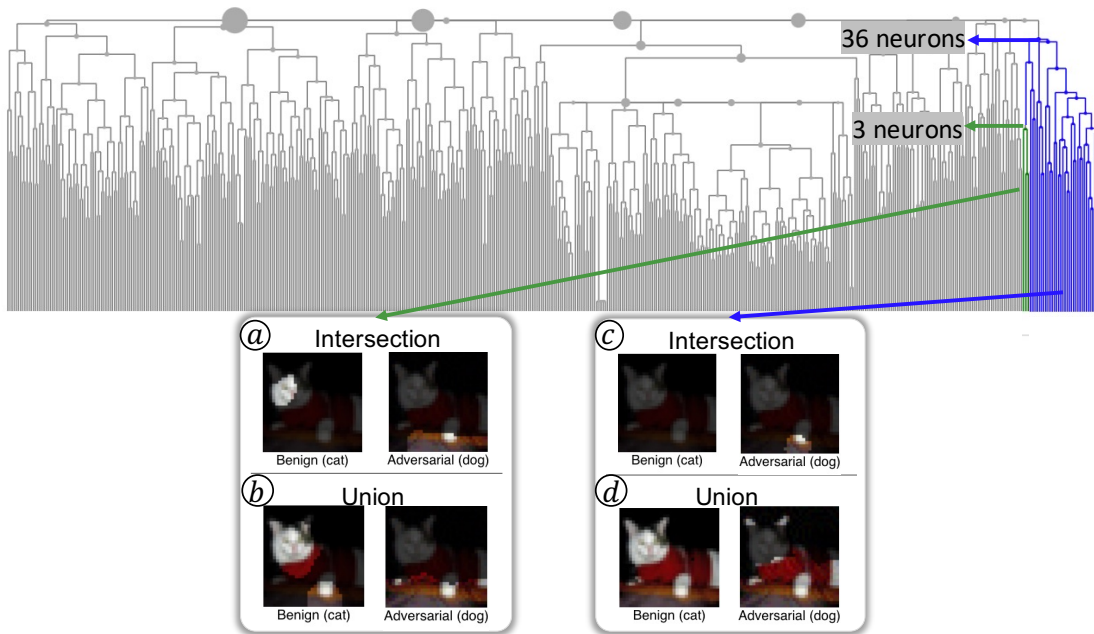


Figure 3.18: Bottom-up exploration of the *Neuron Cluster View* with the **cat** image. When the number of neurons within cluster is increased, the behavior of the neurons on the benign image becomes more and more varied, whereas their behavior on the adversarial image is still similar (e.g. missing the cat face region.)

Exploring Neurons with a Hierarchy We present our explorations with the neurons for the **cat** image in Figure 3.13d and the **plane** image in Figure 3.17 to demonstrate how the *Neuron Cluster View* can help efficiently explore a large number of neurons. The 512 neurons for the **cat** image are explored in a bottom-up order, whereas the 512 neurons for the **plane** image are explored in a top-down order.

Bottom-Up Exploration. In the previous analysis on **cat** misclassified as **dog**, neuron

403 is inspected and interpreted. In the *Neuron Cluster View*, this neuron is highlighted with red, as seen at the right side of Figure 3.13-e. Here, the dendrogram is generated based on the similarities of each neuron’s benign and adversarial RFs. To quickly identify neurons that behave similarly with neuron 403, we locate clusters to which the highlighted neuron belongs.

As shown in Figure 3.18(top), we select the small green cluster consisting of 3 neurons and examine the intersection and union of RFs. Through the intersection (Figure 3.18-a), we notice that all 3 neurons capture the **cat**’s face region in the benign image and the **cat**’s feet region in the adversarial image. In Figure 3.18-b, the union of RFs indicates that the neurons’ behavior varies a little in the benign image, but every neuron captures the feet region in the adversarial image.

We move two steps up and select the bigger blue cluster, which contains 36 neurons. This time, the union of RFs covers almost all of the image (Figure 3.18-d), whereas the intersection of RFs covers nothing of the image (Figure 3.18-c). This indicates that the behavior of neurons on the benign image has more variation within the cluster. However, for the adversarial image, while the intersection of RFs still captures the feet region, the union of RFs covers more area (mostly on the lower half of the body) but still does not cover the face region. This indicates that all neurons within the cluster try to capture the feet region and ignore the face region. Therefore, although these neurons may capture different features of the benign image, they are altered to capture similar features of the adversarial image.

From these two different cluster selections, we observe that, indeed, there exist neurons that behave similarly with others. One interesting insight is that the adversarial attack perturbs the images so that many neurons (36 neurons) act to make similar mistakes (i.e., missing **cat**’s face and focusing on their feet).

Top-Down Exploration. Figure 3.19 shows the exploration of the neurons for the benign **plane** image. From the dendrogram, the big green cluster immediately draws our attention. This cluster has 95 neurons, and all capture very similar image features, which is indicated by the very low vertical positions of the tree nodes. Selecting this cluster and exploring the intersection and union of the corresponding 95 RFs (Figure 3.19-a), we see that this cluster includes all the “dead” neurons, i.e., they all capture nothing from the **plane** image. These neurons can be excluded from further analysis. We also explore several other big clusters to explore their correspondence to the image features. For example, the red cluster of 28 neurons

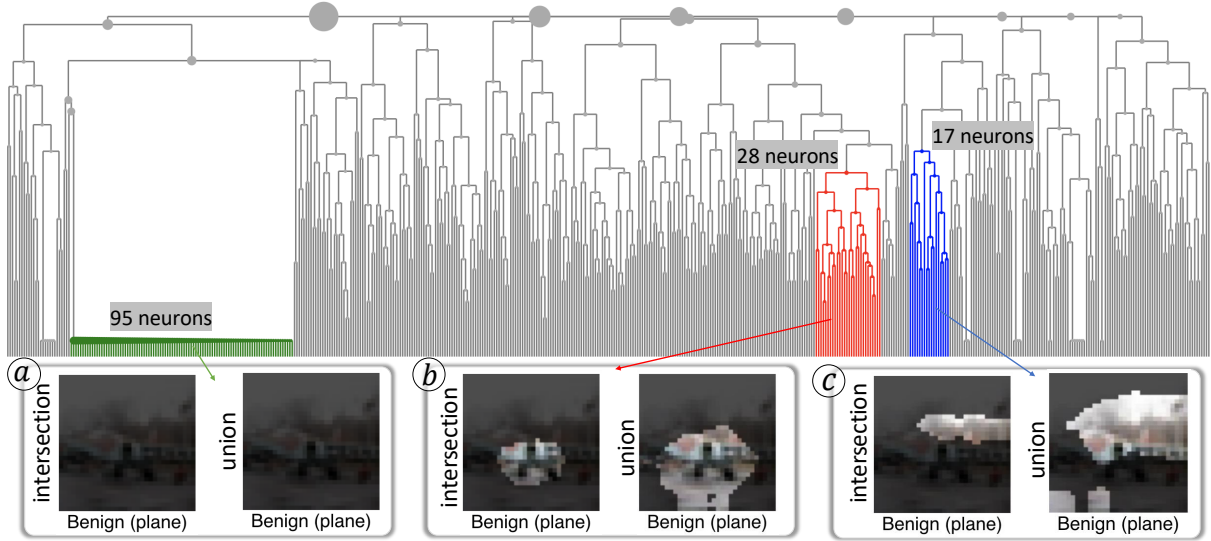


Figure 3.19: The *Neuron Cluster View* when exploring the `plane` image. Through this exploration, the distribution of features can be roughly summarized. For example, the green cluster of neurons (a) capture no features and thus can be ignored during single neuron exploration. The red cluster (b) captures mostly the body and wheel region of the plane, while the blue cluster (c) captures the sky above the plane.

all capture the body and wheel region of the `plane`, as shown by the intersection of their RFs (Figure 3.19-b). Meanwhile, individual neurons also extract some sounding features, as shown by the union of their RFs. For the blue cluster of 17 neurons, the common region is the sky above the `plane` (Figure 3.19-c). Some neurons also extract the lower-left corner of the image, making the functionalities of neurons in this cluster a little more diverse, which is indicated by the higher vertical positions of the tree nodes. Exploring the big clusters only can already provide an overall picture of how image features have been distributed across the neurons. For any clusters, users can interactively traverse the dendrogram from a parent to its child to narrow down the analysis focus.

3.2.5.2 Places365 Dataset

We now demonstrate an analysis case with the Places365 dataset [161]. We use the validation set as the inputs, which contains 100 images for each of the 365 scene categories. The CNN we study is a pretrained wide ResNet18 model [163]. Overall, 19,745 out of 36,500 images are predicted correctly, while 12,298 images of the 19,475 are successfully attacked. Since the number of classes (365) is much larger than the CIFAR10 dataset, we directly select images in the backend and lay emphasis on the neuron space exploration.

We select an image and its adversarial counterpart, where the benign image’s category is

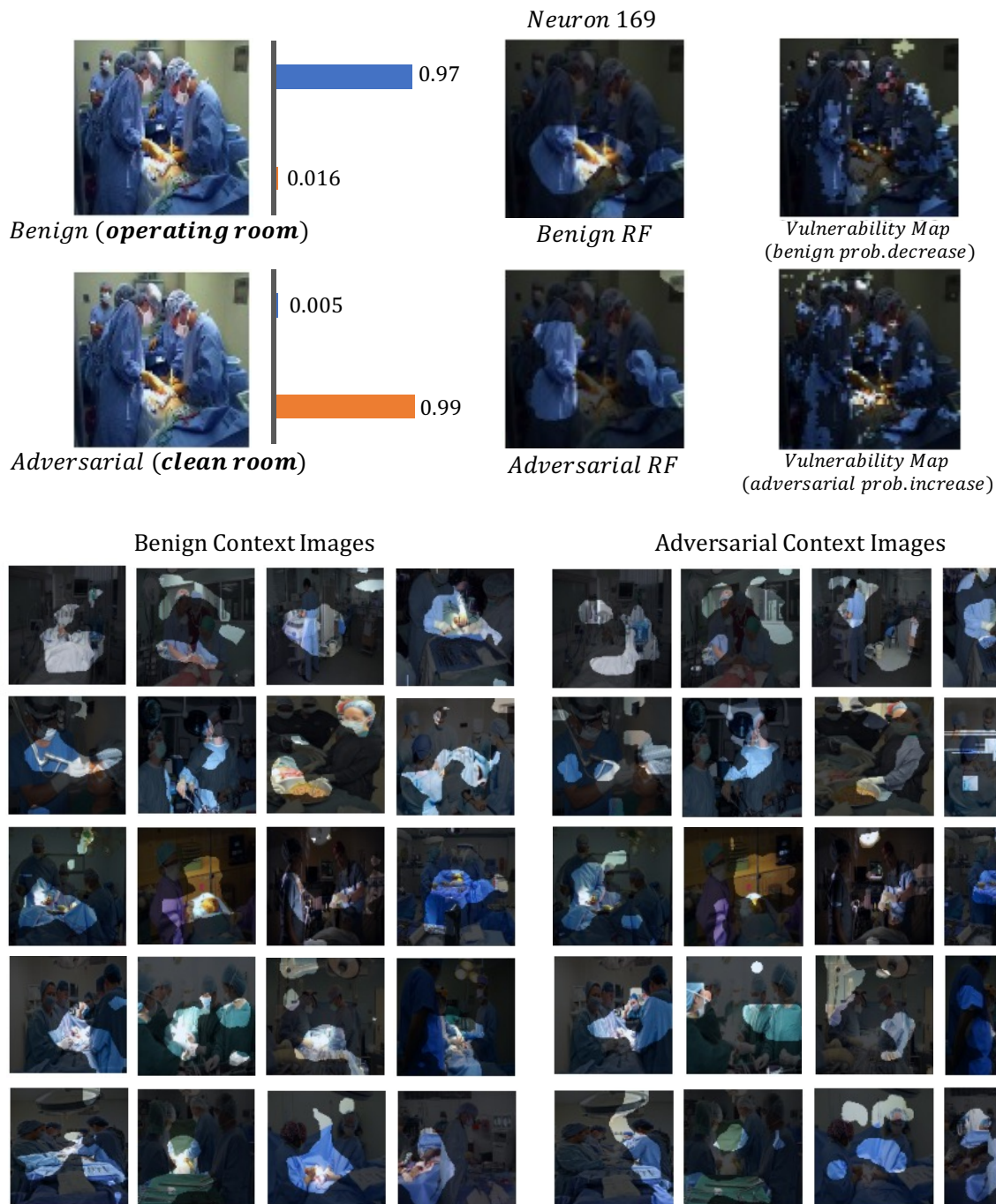


Figure 3.20: Demonstration of a case study with the Places365 dataset. For the selected neuron, the operating table and surgeons' hands are mostly captured in the benign image, but the surgical gowns (blue or green colors) are mostly captured in the adversarial image. This finding can also be verified through the context images.

operating room, but the adversarial image is misclassified as `clean room`, as shown at the top-left corner of Figure 3.20. This image is selected because it is among the images that change the prediction probabilities the most. In addition, a large number of `operating room` images

are misclassified as `clean room`, so it is meaningful to summarize the attacking patterns of this group of images.

As shown in Figure 3.20, although there is little or no observable differences between the benign and adversarial images, the CNN’s probability outputs are significantly different. We examine the vulnerability maps presented on the top right of Figure 3.20. Through the highlighted area, we observe that perturbations around the surgical gowns decrease the benign probability the most, whereas perturbations around the operating table and hands region increase the adversarial probability the most.

We then select neuron 169 for further exploration, which is one of the top neurons with the largest overlap between the RF and the vulnerability map for benign probability decrease. From the RFs at the top middle of Figure 3.20, we can see a clear difference in the RFs of benign and adversarial images: the neuron focuses more on the hands and operating table regions in the benign image but focuses more on the surgical gown regions in the adversarial image.

We then examine the RFs of the benign and adversarial context images, which are listed as grids in Figure 3.20 (bottom). Similar to the selected image, most benign context images highlight regions around the hands and operating table, while most adversarial context images highlight regions around the surgical gowns.

Insight. It can be concluded that the perturbations around the operating table decline the `operating room` feature, while the perturbations around the surgical gowns make neuron 169 shift its focus to the gown feature, which is also one component of most true `clean room` images. In this way, the adversarial attack succeeds in alternating the neuron’s behavior.

3.2.6 Domain Experts’ Feedback

We conducted case studies with two domain experts (E_1 and E_2) working on various types of adversarial attacks, e.g., adversarial examples, data poisoning, and adversarial code analysis. Both are full-time research scientists from a research lab and hold a Ph.D. degree in computer science (with a focus on adversarial deep learning). The study lasted for ~ 90 minutes for each expert, in which we explained our two targeted research questions, guided the experts to explore the cases we explained in Section 3.2.5, and paused at the findings that triggered the experts for comments. Lastly, we conducted think-aloud discussions and open-ended interviews with the experts to collect their feedback. Neither expert co-authored this work (for objective evaluations).

In general, both experts provided positive feedback on the way we answer the two research

questions and the interactivity of our coordinated visual analytics system. E_1 supported our choice of focusing only on the last convolutional layer of a CNN for analysis. The high-level features extracted by neurons of this layer help to take advantage of the human-understandable semantics for interpretation. Also, our vulnerability map was very inspiring to him after seeing the results in Figure 3.17-l. The observation motivated him to fine-tune the CNN by masking the wheel regions (i.e., the vulnerable pixels) and enforcing the CNN to learn other **car**-related features to improve the model’s robustness. This idea is along the same lines as the recent work of Ilyas et al. [164]. E_2 enjoyed the coordination between different views of our system. He commented that their colleagues work more with different Python code snippets to generate visual evidences for interpretations. However, the discrete visual reasoning process significantly hindered their analysis efficiency. Our integrated and coordinated visualizations (e.g., cross-space explorations) provided a more friendly interface to involve human in the analysis process. During the study, E_2 also had many intriguing thoughts triggered by our visual interpretation results. For example, he suspected the non-robust features between two classes (e.g., wheels for **car** and **plane** in Figure 3.17) could be robust features between another two classes (e.g., **car** and **cat**). He also suggested pasting the wheel features from a **car** image to a **plane** image to probe the CNN’s behavior. Furthermore, he seconded that our hierarchical exploration of neurons helps to efficiently discover the “dead” neurons or neurons with similar functionalities, which could be used to prune CNNs and compress them.

There are also several limitations and potential improvements suggested by the experts. First, E_1 commented that PGD does not guarantee to attack continuous image features and suggested considering the synergetic effects of multiple image features when perturbing an image to generate its vulnerability maps. Second, E_2 believed that different attacking methods (e.g., PGD or FGSM [37]) may result in different vulnerability levels of the neurons. But, he also speculated that the neurons’ vulnerability should not change too much when attacking the same CNN. This is a good task for us to validate and compare different adversarial attack methods in the future. Third, E_2 also talked about how to extend our work to other adversarial attacks, e.g., the attacks on graph data or source code, where (unlike images) the data instances may not carry much semantics. How to visually distill human-understandable semantics and use them to interpret adversarial attacks on other types of data is an interesting future direction for us to explore.

3.2.7 Discussion, Limitations, and Future Work

There are several *limitations* in our current work. *First*, as pointed out by the experts, we perturb one image patch at a time when generating the vulnerability map (Figure 3.14-a), and did not consider the combined effects of multiple patches (as there are so many possible combinations). This is a common limitation of similar saliency map generation algorithms, e.g., [154]. A potential remedy for this issue is to randomly sample some patch combinations during the perturbation process and evenly attribute the probability changes to the sampled patches. *Second*, as we extend our study to the Places365 dataset, we notice the scalability issue of our current system. Specifically, our image space exploration component (Figure 3.13-a) is not capable of handling hundreds of classes. In practice, however, as the exploration usually focuses on a small number of classes, we plan to ask users to provide their classes of interest before the analysis and load the *Prediction Matrix* for the corresponding classes only. This would also narrow down the number of data instances to be explored. *Lastly*, our current evaluation relies on expert users with sufficient knowledge of adversarial attacks. In the future, we intend to recruit more users with different expert levels to thoroughly evaluate the usability of different visualization components.

Regarding the *generalizability* of our work, we want to emphasize three points. *First*, our proposed solutions are applicable to both targeted and untargeted attacks. In fact, based on the instances' distribution from the *Prediction Matrix View*, we can even infer if the studied attack is targeted or not (i.e., the cells for the targeted class will have darker color). *Second*, our way of identifying the vulnerable neurons and their follow-up interpretations are agnostic to the attacking algorithms. We focused on PGD in this work, as it is the most popular one, but there are no attacking-specific features preventing us from applying our system to other attacks, such as FGSM. *Third*, our work focuses on adversarial attacks in CNNs only, but we believe our general idea of identifying critical neurons to an attack through perturbation is applicable to other types of data (e.g., texts or graphs) and other deep learning models.

In this work, we focus on understanding adversarial attacks through *Instance-based Analysis*. In the future, we aim to provide visual analytics on the collective behavior of clusters of neurons or input images. However, there are two reasons why this work provides sufficient information for understanding neuron behavior and model vulnerabilities. First, instance-based explanation is fundamental for concrete and intuitive analysis. The collective pattern of a clus-

ter is usually approximated as the instance behavior of a representative, and instance-based analysis is necessary when the object of study is the abnormalities. Second, we have achieved multi-level analysis through several techniques. To explore behavior of groups of neurons, we provide the *Neuron Cluster View* and visualize the intersections and unions of their RFs. To explore by groups of images, we present the sorted context images in the *Receptive Field View* to support the findings within the selected image. Finally, this work presents multiple cases with two different datasets in Section 3.2.5 to prove the reproducibility and generalizability of the analytic workflow.

System Interactivity. The most time-consuming part of our system is to generate the vulnerability map for an image, as it requires to perturb all possible image patches with numerous forward-pass of the studied CNN (line 4-5 in Algorithm 1). The stride parameter (i.e., the variable s in Algorithm 1) also influences the performance. Currently, we run the system on a CPU machine and set the stride to be 1. Generating the vulnerability map is ~ 45 seconds. This latency did not significantly impact our analysis, because (1) the focused image is not updated very often, so as to the computation of its vulnerability map, (2) Algorithm 1 is embarrassingly parallel and the CNN’s forward-pass time can be significantly shrunk if we use a GPU-enabled machine, (3) we can also easily adjust the stride to balance the trade-off between accuracy and latency. For example, increasing the stride to 2 will not much affect the appearance of the resulting vulnerability map, but will shrink the computation time to $1/4$. As the interactivity of a visual analytics system is a crucial factor for maintaining users’ mental map during analysis, comprehensively profiling the latency of our system with different CNN models on different hardware platforms is also part of our planned future work.

3.2.8 Conclusion

In this work, we introduce a visual analytics approach to understanding the adversarial attacks happened in CNNs. Our approach targets to answer two questions of what neurons are more vulnerable and how they have worked in the prediction process. By perturbing the input images or neuron activations, we decompose the consequence of adversarial attacks into individual neurons and introduce several metrics to rank the neurons by their vulnerability. Hierarchical clustering is used to structurally explore the excessive number of neurons and reveal their behavior similarity in benign/adversarial images. Integrating all these technical innovations, we design a visual analytics system, empowering users to conveniently explore the input images and CNN

neurons jointly, and use the semantics carried by input images and extracted by CNN neurons to interpret adversarial attacks. Through multiple case studies and domain experts' feedback, we validate the efficacy of our visual analytics approach.

Chapter 4

Visual Analytics and Machine Learning Interpretability for Model Improvement

The final stage in iterative ML research is model improvement, a crucial process aimed at refining a model's performance by addressing its identified limitations. Once a model's initial behaviors and patterns have been thoroughly analyzed, researchers shift their focus to enhancing its robustness, accuracy, and adaptability across varied applications. This stage is especially important because it involves optimizing performance metrics as well as advancing critical criteria such as transparency, reliability, and ethical integrity. By moving from understanding a model's weaknesses to actively enhancing its capabilities, model improvement transforms ML systems into more trustworthy and adaptable applications.

The process of ML improvement relies heavily on understanding the model's inner mechanisms through VA and ML interpretability. **First**, these tools enable a comprehensive assessment of the model's limitations by identifying specific areas where its performance may falter or diverge from expected outcomes, providing a structured overview of its weaknesses. **Second**, interpretability techniques allow for a more in-depth diagnosis of the model's behavior, revealing underlying patterns and potential sources of error that can inform the design of targeted improvements. **Finally**, interpretability and VA techniques support a human-centered approach to model refinement, allowing for fine-grained, interactive improvements guided by domain knowledge. Together, VA and ML interpretability play a critical role in the model development phase, not only by clarifying limitations and guiding targeted improvements but also by fostering a more transparent and trustworthy enhancement process.

Building on this correlation between ML improvement, ML interpretability, and VA, I fo-

cus on enhancing multi-modal foundation models to improve visual comprehension of large-scale image datasets. First, by analyzing cross-attention mechanisms within transformer-based models, I reveal connections between model-generated captions and specific image features, shedding light on biases embedded in the visual data. Second, visual analysis of less-attended image features helps to uncover gaps in caption coverage, identifying weaknesses in the model’s ability to represent relevant image features. Finally, I developed an interactive interface that empowers researchers to dynamically steer the caption generation process to mitigate these identified weaknesses. This integrated system, combining both visual and textual analysis, was validated through case studies with domain practitioners, demonstrating the effectiveness of a user-centered, interpretability-driven approach to enhancing foundation models. This chapter presents the analysis techniques and visualization design in details.

4.1 Visual Analytics for Efficient Image Exploration and User-Guided Image Captioning

Recently, the rapid advancements of large-scale ML models have been mirrored by the success of pre-trained language-image models, such as ALBEF [165], CLIP [166], and BLIP [167]. These models take a large number of image-text pairs (i.e., images and their captions) as input and learn the correspondence between the two data modalities. Pre-trained language-image models can be used in many downstream tasks, such as zero-shot image classification, image captioning, and visual question answering. The profound potential of these models opens promising avenues in the realm of multi-modal learning.

Borrowing the power of these pre-trained language-image models, this work employs visual analytics to approach two common challenges relating to the exploration of image datasets and the quality of generated captions. **First**, before using an image dataset to train ML models, domain practitioners often need to obtain an efficient overview of the dataset and be aware of potential data-related issues. For instance, Ribeiro et al. [168] illustrated how a **husky dog** image was misclassified as a **wolf** due to the presence of snow in the image’s background. This happened because, in the training set of **wolf** images, snow was a consistently co-occurring feature. It led to the model developing a bias, assuming any image with snow to be a **wolf**. Preventing

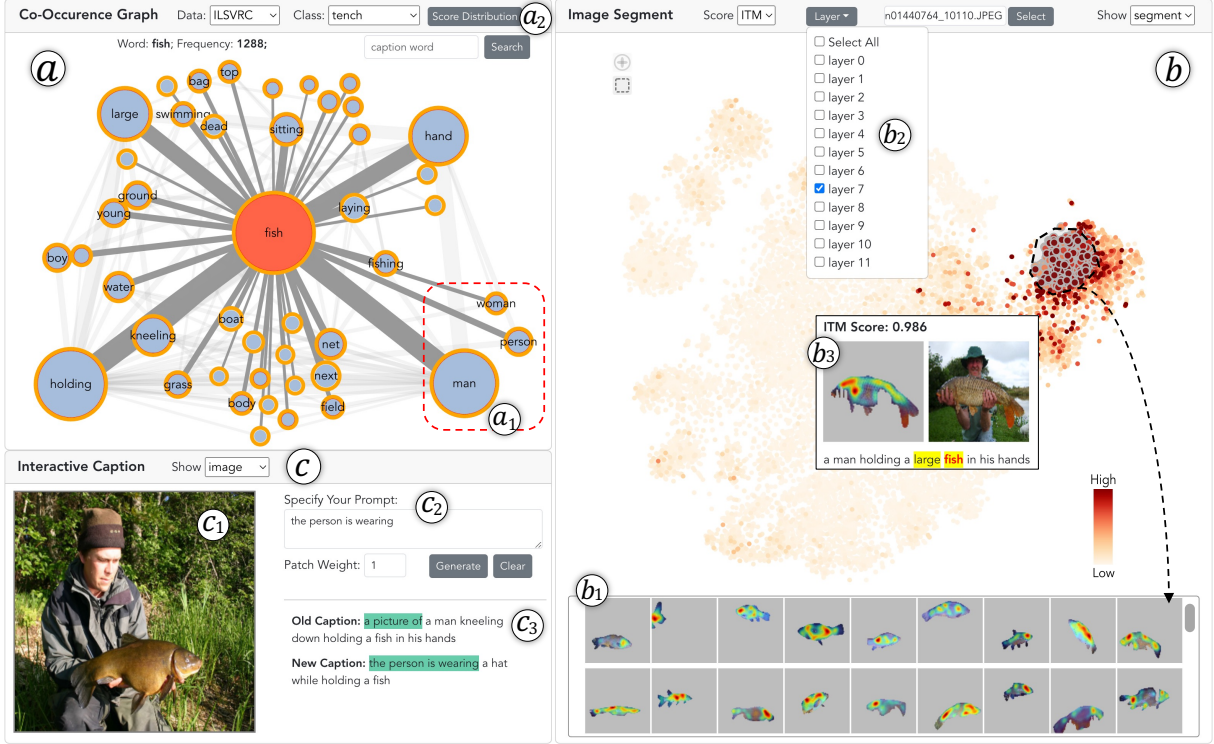


Figure 4.1: Our system contains three main views. (a) The *Co-Occurrence Graph* employs a node-link diagram to present the dominant and co-occurred features of an image dataset. (b) The *Image Segment* view uses a scatterplot to lay out image segments and color them based on the attention they received. (c) The *Interactive Caption* view provides an interface to steer the caption generation process.

such abnormal model behavior necessitates efficiently exploring large-scale image datasets and recognizing possible data biases. The pre-trained language-image models can generate captions that textualize visual features in image datasets and enable their efficient exploration. However, such image captioning models are not without imperfections and there are cases where certain image features are not covered. For example, the caption of an image with a **cat** and a **dog** may only include descriptions of the **dog** due to various reasons. However, the **cat** may be of more interest to users. Recognizing such cases and summarizing their common patterns are vital for understanding the models' captioning behavior, evaluating the models, and optimizing their usage. This pressing need, however, is notably impeded by the *second* challenge, i.e., it is arduous to manually scrutinize individual captioning cases, collect them, and extract their common patterns.

Due to the widespread nature of these two challenges, existing studies have proactively attempted to address them. For the first one, DendroMap [169] organizes a large-scale image

dataset in a treemap based layout, allowing users to grasp a quick overview of an enormous number of images. Focusing on tabular data, CoFact [170] generates counterfactual examples for data instances with selected features and compares the distributions of the two sets of data to shed light on bias-related data issues. Nevertheless, gleaning insights from these works requires substantial manual effort and data exploration. For the second challenge, various quantitative metrics have been proposed to measure the quality of generated captions. For example, when ground-truth captions are available, BLEU [171] and CIDEr [172] are frequently utilized to assess the caption quality. However, these metrics are often overly aggregated, providing limited insights into where the weaknesses of a captioning model reside. Furthermore, in cases where ground-truth captions are not available or users wish to generate captions for specific image features, these metrics prove inadequate.

In this chapter, we introduce a visual analytics solution to approach these two challenges. For the first one, we harness the capabilities of pre-trained language-image models, i.e., BLIP [167], to generate captions for large-scale image datasets. These captions textualize the image content with descriptive words. We then extract key words out of the captions and build a node-link diagram to demonstrate them and their co-occurrence frequency. The diagram provides an efficient summary, guiding a detailed exploration of image features. The feature co-occurrence from the diagram further reveals potential data biases. For the second challenge, we decompose images into semantic image segments and reflect their associations with different words from the captions, utilizing the cross-attention of BLIP. By grouping the image segments and depicting their association strengths with different words, we reveal the dominant and less-covered image features. Additionally, we also provide an interface for users to interactively emphasize the overlooked visual features and steer the caption generation process. All the above functionalities have been integrated into a coordinated visual analytics system with three main components (Figure 4.1). We use the system to explore two real-world image datasets (ImageNet [173] and COCO [174]) and share insights that can be gleaned specifically from our system. In summary, this work’s contributions include:

1. An approach to effectively exploring large-scale image datasets and disclosing potential data bias rooted in the datasets.
2. A visual analytics approach that intuitively presents the association between texts and images, revealing the weaknesses of pre-trained language-image models in visual feature coverage.

3. A human-in-the-loop solution to steer the existing caption generation process, generating descriptions for desired visual features.

4.1.1 Related Work

Our work is intricately linked to the exploration of large-scale image datasets, the diagnosis and steering of ML models, and the interpretation of transformers’ attention. These topics are reviewed as follows.

4.1.1.1 Large-Scale Image Data Exploration

Exploring and comprehending large-scale image datasets fall within the purview of *data-centric AI*, which improves ML models by better understanding the training data and enhancing their quality [175, 176]. In visual analytics, there has been a considerable amount of work focusing on efficient image exploration. These endeavors can generally be categorized into two groups, exploring images using their performance from ML models or their predominant features. For the first group, Squares [34] encodes image instances as stacked boxes and analyzes them based on their prediction scores. Blocks [177] groups images by their original and predicted labels, and presents the under-performing groups in a hierarchical confusion matrix. SliceTeller [178] generates image subsets (slices) based on annotated metadata, and focuses on analyzing under-performing data slices. For the second group, EmbeddingProjector [179] presents images through dimensionality reduction algorithms and scatterplot visualizations. Similarly, Karpathy [180] employs t-SNE to project a large number of images and uses a grid layout to arrange them to minimize overlaps. Bäuerle et al. [181] use a honeycomb representation to hierarchically organize images, supporting meta-data comparison among image groups. DendroMap [169] adopts an interactive tree-map to facilitate multi-level image explorations and unveil potential data biases within the images. Targeting to enrich image exploration with semantics, Xie et al. [182] adopt a forest-based co-embedding method to bind words with images. In contrast to these approaches, our work transforms images into the text domain through image captioning and conducts analyses on the texts to provide insights into the original images. Our solution does not rely on pre-existing performance from ML models or pre-defined image attributes. Also, the modality transformation from images to texts can be largely automated and the text analysis is less labor-intensive compared to existing image exploration approaches.

4.1.1.2 ML Model Diagnosis and Steering

Model diagnosis is the process of reasoning on unexpected model behaviors and/or uncovering model weaknesses [183]. For example, Seq2seq-Vis [93] dissects the execution of a Seq2Seq model into five stages and debugs potential issues at individual stages through customized visualizations. DQNViz [85] employs a series of chart visualizations and pattern-matching algorithms to visually summarize the behavior patterns of a reinforcement learning agent. These patterns are instrumental in unveiling the agent’s playing strategy and diagnosing instances where it fails. AEVis [184] analyzes CNNs’ weaknesses against adversarial attacks by comparing the models’ behavior on benign and adversarial instances. In this work, we diagnose pre-trained language-image models to reveal their weaknesses in image captioning. By summarizing the coverage of generated captions over image features, we provide a new angle to assess the captioning models. *Model steering* [183] empowers users to directly engage with an ML model and control its execution behavior by injecting human knowledge. For example, Yang et al. [185] proposed ReVision to interactively construct hierarchical clustering trees. In the construction process, users can infuse both public and private knowledge to interactively supervise the tree hierarchy. Ming et al. [186] used representative prototypes to interpret ML models. Their ProtoSteer system enables users to interactively add, remove, or modify prototypes to enhance interpretability. More model steering examples also include [187, 188, 189, 190, 191, 192]. In this work, we provide an interface to steer image captioning models. The interface allows users to dynamically emphasize different visual objects within images. The promptly generated captions also provide users with immediate feedback to iterate the generation process.

4.1.1.3 Visualization of Transformers’ Attentions

The attention mechanisms [91], especially their application in transformers [72], have achieved great success in the past decade, yielding numerous powerful language [73, 193, 194] and vision [76, 101] models. Meanwhile, various techniques have been utilized to visualize attentions. The most popular ones include flow maps [92, 93], parallel coordinates plots [89], and heatmaps [90, 80, 94]. For example, Section 2.2 uses dimensionality reduction, heatmaps, and scatterplots to effectively summarize attention patterns of vision transformers (ViTs). These patterns then helped them diagnose different behaviors of the ViTs. Park et al. [80] used heatmaps to distinguish attention patterns in a language transformer and assist the probing of *query* and *key* features within each attention head. Jaunet et al. [90] explained a visual question-answering

transformer through attention heatmaps, exploring potential biases within it. Yeh et al. [195] analyzed different heads’ attention behavior by jointly performing dimensionality reduction on the *query* and *key* embeddings. The interpretation of attentions can also be used to compare different attention heads [196] or ML models [81]. We interpret the cross-attentions of language-image models and use them to effectively explore one data modality (images) through the other (texts). Our interpretation also helps to disclose the weaknesses of the language-image models.

4.1.2 Background (BLIP and SAM)

Bootstrapping Language-Image Pre-training (BLIP) [167]. Although there are many pre-trained language-image models, we focus specifically on BLIP [167] in this work, as it is one of the latest and represents the state-of-the-art (SOTA) till the beginning of this work. In Figure 4.2, we explain two functions that BLIP can perform, which are a subset of its capabilities. To get full details of the model, we refer readers to [167].

- The first one is caption generation, which is administered through a language modeling (LM) loss during training (Figure 4.2a). Given an image, the visual encoder of BLIP, a ViT, first decomposes it into p^2 patches and encodes each into a d -dimensional vector. The resulting $p^2 \times d$ matrix is then passed to the text decoder (the LM part) for caption generation. Besides the patch embeddings, the text decoder also takes a prompt (e.g., “a picture of”) as the beginning of the generated caption and it performs next token prediction recursively until reaching the termination criteria.
- The second function is image-text matching (ITM, Figure 4.2b), which outputs a probability value indicating if an image and a caption match with each other. Through the ViT, the image is encoded into p^2 d -dimensional embeddings. Meanwhile, the caption (with t words) is also encoded into embeddings through a bi-directional self-attention component. The cross-attention between the embeddings of the p^2 patches and t tokens is then learned, resulting in a $p^2 \times t$ matrix. Each element of this matrix reflects how strongly a word token attends to an image patch. This cross-attention matrix is our analysis focus while associating words and image patches.

Pre-trained over hundreds of millions of image-text pairs (either human-annotated or web-crawled), BLIP has demonstrated strong generalizability in many downstream tasks. We use its pre-trained parameters for the above two functions in this work.

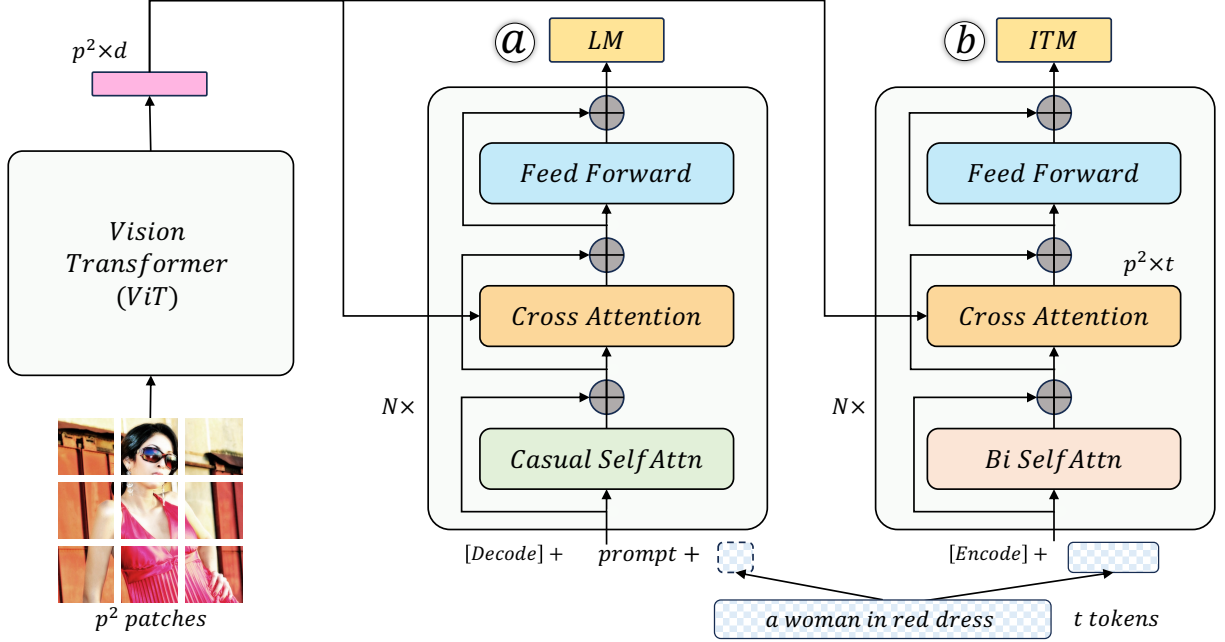


Figure 4.2: BLIP for (a) image captioning and (b) image-text matching.

Segment Anything Model (SAM) [197]. Our work employs SAM to extract semantically meaningful segments from images. SAM is pre-trained on a carefully curated dataset (i.e., SA-1B [197]) with 11 million images and over 1 billion segment masks. It can be readily applied to new datasets beyond its original training images. We use this model as a black box, where the input is an image and the outputs are multiple segments of the image, each is denoted as a binary mask indicating what pixels belong to the segment and what pixels do not. Besides SAM, we have also considered using region proposal or superpixel algorithms [141, 198, 199] to decompose an image. However, compared to SAM, those methods either more severely break the image semantics or contain more overlaps in the resulting segments. Thus, we chose SAM, which is also the SOTA method for image segmentation.

4.1.3 Goals and Tasks

By reviewing the literature and discussing with domain practitioners on pre-trained ML models, we identified several pain points when model designers train, evaluate, and use these models. For example, Bertucci et al. [169] emphasized the importance of comprehending images before using them for model training. Gou et al. [43] detailed the need to assess ML models to reveal their weaknesses. Multiple works [185, 186, 190, 188] demonstrated the importance of steering ML models.

4.1.3.1 Goals

Our review and discussions identified three primary goals:

- **G1: Efficiently explore large-scale image datasets and expose potential data bias.** Prior to training an image classifier (or other ML models), it is customary for domain practitioners to develop a comprehensive understanding of the image dataset and be mindful of potential issues associated with the dataset. For instance, as demonstrated in [168], `snow` often presents in the training images of `wolf` as background. This undesired feature co-occurrence can mislead trained ML models into associating the existence of `snow` with `wolf`, a correlation that is not always held. Consequently, the classifier might mislabel a `dog` image as `wolf` as long as `snow` is detected. However, due to the substantial volume of images, manually scrutinizing each image and extracting a comprehensive overview of the dataset poses a considerable challenge.
- **G2: Reveal image features/objects that are not well covered by the generated captions.** Despite the remarkable captioning performance exhibited by pre-trained language-image models, there remain cases where the models fail to generate captions for specific visual features. For instance, in Figure 4.2, the sunglasses were not in the generated caption, despite being conspicuous features. By collecting these cases and summarizing features that frequently receive inadequate coverage, it becomes possible to unveil behavioral tendencies and potential limitations of the models. These limitations might encompass a deficient grasp of associations between certain visual traits and linguistic expressions. However, simultaneously examining numerous images along with their captions, while also identifying the inadequately covered features, proves to be an exceedingly complex endeavor.
- **G3: Steer caption generation with human knowledge.** Another pain point faced by users of image captioning models is the limited control over the generation process. Often, the generated captions fail to adequately describe the visual features that are of particular interest to the users. Although those features may have been learned by the pre-trained models, certain constraints, such as caption length, can lead to the omission of the features' description. This occurs because the model deems them less crucial. However, the features that the model deems as important might not align with the features that hold significance in the users' perspective. To mitigate this, we target to enable users to actively specify feature importance and guide the caption generation.

4.1.3.2 Tasks

Through iterative refinements over different design considerations, we have concretized our three goals into the following actionable tasks:

- **T1: Textualize image contents and reveal the dominant features and feature co-occurrence.** Thanks to the remarkable performance of image captioning models, the generated captions can capture the main contents of images very well. A holistic overview of an enormous number of images can therefore be obtained from the captions without manually eyeballing individual images. The exploration of captions is also much easier than images, as they can be easily broken down. The appearance frequency of different words in the captions and their co-occurrence frequency effectively reflect the dominant and co-occurred image features. Accomplishing this task can substantially support **G1**.
- **T2: Extract visual features of images, explore them by their semantic similarity, and associate them with texts.** An image may come with diverse features, e.g., a photo of a park may include several persons, their pets, the sun, and trees. An effective overview of an image dataset and its dominant features, therefore, requires to extract those features and project them by their similarities to augment the data patterns (**G1**). The attention that different features receive when generating a word of the captions can also be used to associate the image features with caption words, which prepares for **T3**.
- **T3: Coordinately explore the association between texts and visual features.** A mapping between individual caption words and extracted image features is always necessary when exploring images through their captions or vice versa. This task enables the mutual exploration between the two data modalities (**G1**). The correctness and strengths of the association will also reveal the quality of the pre-trained ML model (**G2**), disclosing its potential weaknesses.
- **T4: Empower users to steer caption generation by interacting with prompts or visual features.** To achieve **G3**, an interaction mechanism is needed to allow users to manipulate the settings before caption generation, engage with different image features, and compare the generated captions for feature verification. This task aims to grant users greater control over the caption generation process, empowering them to craft captions that align with their preferences.

4.1.4 Methodology and Visual Analytics System

We explain our methodology in the following three subsections, each presents the details for achieving one of the three goals.

4.1.4.1 Efficient Image Data Exploration

Our first goal is to efficiently explore large-scale image datasets (**G1**). We believe image captions generated using language-image models can help here, as they can very well describe the contents of images (**T1**).

Figure 4.3 shows an overview of our approach. Given a large-scale image dataset, we first feed individual images into a pre-trained language-image model for caption generation (Figure 4.3a). Specifically, the model here is the ViT and LM part of BLIP (Figure 4.2). The generated captions capture the essence of the images. An effective exploration of them can then provide insights into the original images without manually eyeballing individual ones. The transformation from images to texts is notably less labor-intensive, as the captioning process is mostly automated. Moreover, exploring those captions is more convenient compared to handling the images directly, as they can be easily deconstructed (into words with inherent semantics) and analyzed.

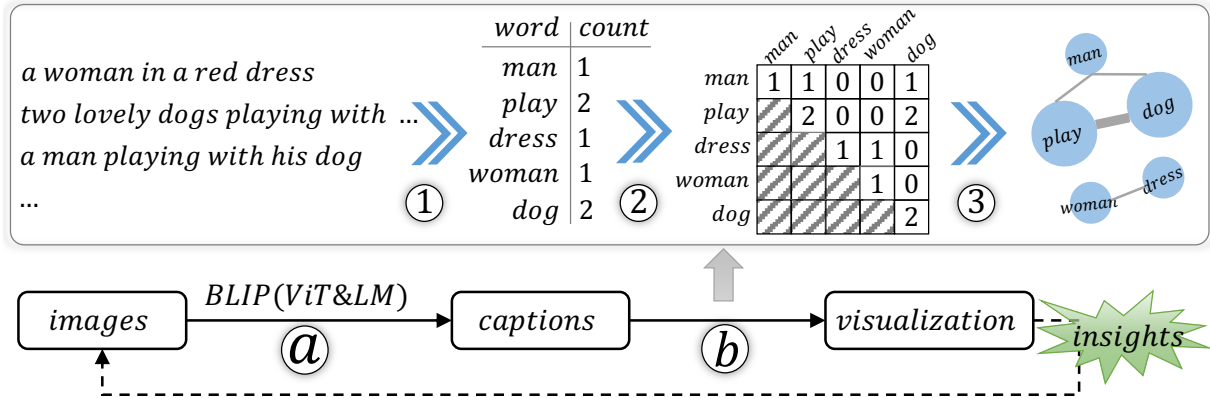


Figure 4.3: Exploring images through our caption *Co-Occurrence Graph*.

We build a node-link diagram to effectively explore the generated captions (Figure 4.3b). The building process can be described as follows. **First**, we tokenize the captions into words and standardize the same word in various formats (e.g., converting plural nouns into their singular form with TextBlob [200]). Using NLTK [201], we remove stop words, such as ‘a’, ‘the’, and ‘with’, as they carry no semantics. For the remaining unique words, we count their appearance in the captions. **Second**, a co-occurrence matrix is constructed, each cell denotes the count that

the words from the corresponding row and column occurred simultaneously in one caption. The resulting matrix will be symmetric and we only need half of it. **Third**, a node-link diagram is built from the co-occurrence matrix. Each node represents a unique word and its size reflects the count that the word occurred in all captions (i.e., its frequency). A link between two nodes indicates the corresponding words co-occurred in at least one caption. Its width reflects the co-occurrence frequency. We use square root mapping and size clipping to handle too small/large nodes, as well as too thin/thick links.

The node-link diagram is visualized using the force-directed layout in the *Co-Occurrence Graph* view of our system (Figure 4.1a). Dragging-and-dropping a node will pin it at a fixed position. The represented words are centered in each node and we hide the words in smaller nodes to reduce text clutter. Clicking a node will show details of the corresponding word and highlight image segments that have strong cross-attentions with it in the *Image Segment* view (explained later). Nodes and links with very small values can be filtered out to help users focus on the more important nodes and links. Interactively exploring the node-link diagram helps to comprehend the underlying images and features. For example, Figure 4.1a shows the node-link diagram built on top of the captions generated from the **tench** images of ImageNet [173]. From the figure, it is evident that most of the images contain **fish** and **man**, and the two features appear concurrently in most images (reflected by the thick link). Considering the big node for **holding**, we can imagine that most images include the content of “a man holds a fish”.

Additionally, each image-caption pair is also fed into the ITM part of BLIP (Figure 4.2b) to evaluate the caption quality. The histogram in Figure 4.4a (popped up after clicking the “Score Distribution” in Figure 4.1-a2) shows the ITM score distribution of all image-caption pairs. To present the word distribution in a certain score range, we use the outer ring of a node to reflect the portion of the corresponding word in that range. In Figure 4.4a, the brushed range is $[0, 0.6]$, and the outer rings of all nodes in Figure 4.4b show the portion of the corresponding words appearing in these low-quality captions. Specifically, the explored images here are **red_wolf** images from ImageNet and a big portion of the word **horse** appears in low-quality captions. The word also has a connection to **brown**, indicating a confusion between **red_wolf** and **brown_horse**. On the other hand, as the **horse** node is small, the issue is less severe.

Note that we have also considered other design choices, such as a matrix representation of the word co-occurrences. Although previous work has shown advantages of matrix visualization

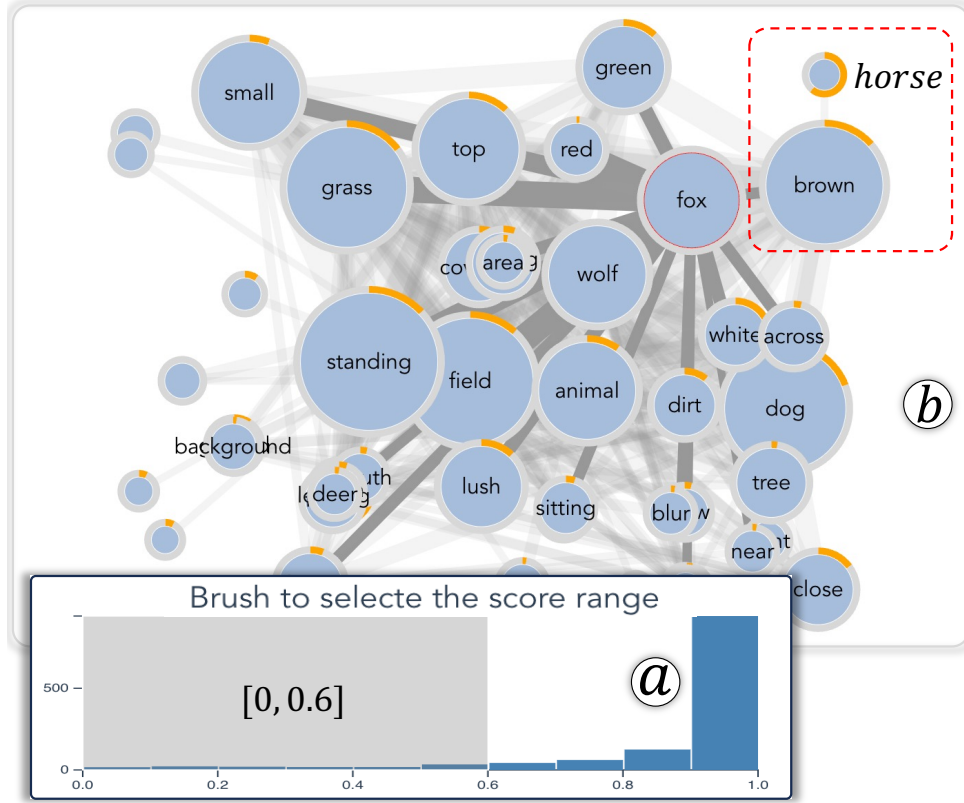


Figure 4.4: The outer ring/arc of each node reflects the portion of the corresponding word in a selected ITM score range ($[0, 0.6]$ in this case).

on basic graph tasks [202], it is less intuitive for complex tasks such as node selection and comparison. Given hundreds of words extracted from image captions, the matrix will contain many rows and columns in the limited screen space. Other design choices include using an arc diagram or a chord diagram to connect the words, but they are also less space-efficient and less intuitive for domain practitioners to interact with.

4.1.4.2 Image-Text Association and Caption Coverage

To accomplish **G2** of revealing caption coverage over image features, we need to first extract meaningful image features (**T2**), and then, disclose their association with different words inside the captions (**T3**).

For Task **T2**, we employ the pre-trained SAM [197] to automatically extract segments out of each image. The resulting segments can be very fragmentary or have big feature overlaps, and thus we perform some filtering steps to distill the most representative ones. Specifically, we first exclude segments that are too small to carry any semantics. These are the segments with sizes less than 1% of the corresponding image. This step significantly reduces the number of segments

by 66%~88% for different classes. Second, we compute the intersection-over-union (IoU) over all pairs of segments from the same image. For any pair, if the IoU is over 0.85, we keep only one out of the two to reduce redundancy. This further reduces the number of segments by 1%~4%. The choices of the two values, 1% and 0.85, are based on our empirical studies, but can be altered on-demand when exploring different datasets.

To efficiently explore and summarize the visual features, we feed all filtered segments into the ViT of BLIP to generate their high-dimensional embeddings. These embeddings are then projected to 2D through a t-SNE+scatterplot visualization (i.e., the *Image Segment* view, Figure 4.1b). Segments representing similar features from different images are close to each other in the 2D space, facilitating the identification of common visual features. Figure 4.11a (as well as Figure 4.1b and Figure 4.8) shows this scatterplot visualization. By lasso-selecting different clusters, we can easily explore similar segments, such as the **fish** and **face** segments in Figure 4.11b and 4.11c, respectively. Clicking a point will pop up a window showing the corresponding segment, the original image that the segment resides, and the generated caption (Figure 4.1-b3).

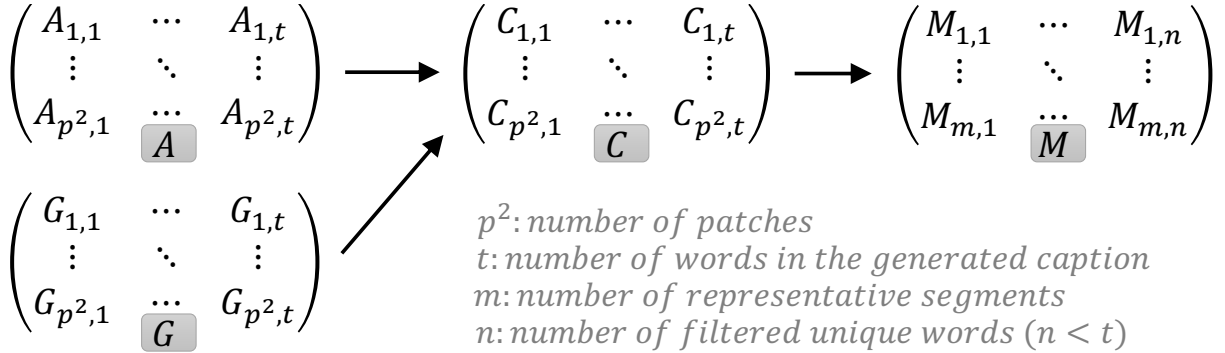


Figure 4.5: The Grad-CAM computation for a single image-caption pair.

For Task **T3**, we associate words with segments through Grad-CAM [203], as it has been widely adopted in early works [165, 204]. Figure 4.5 illustrates the computation process. **First**, for a single image-caption pair, we get its $p^2 \times t$ cross-attention, denoted as A , from the ITM part of BLIP (Figure 4.2b). Each cell of this matrix reflects the attention between the corresponding pair of image patch (row) and caption word (column). **Second**, we get the gradient of A with respect to the ultimate ITM score, and name it as G (i.e., $G_{i,j} = \frac{\partial ITM}{\partial A_{i,j}}$). These two matrices are then merged through an element-wise multiplication to get C , which is the Grad-CAM. C is better than A in reflecting the association between words and patches, as there are cases where

the attention is strong between a patch and a word but the gradient is close to 0. **Third**, we remove the columns of C for stop words and aggregate the rows of C based on how different patches form into segments. The resulting matrix M has a size of $m \times n$, where m is the number of representative segments and n is the number of unique words for the given image-caption pair. Each element of this matrix denotes the attention between a word and a segment. **Lastly**, the computation is extended to all image-caption pairs, and the resulting M from them are joined together by computing the set union of their rows and columns to generate $M^\cup$.

We want to further clarify two details when generating C in Figure 4.5. First, there are multiple copies of A , one for each attention head of the BLIP. In our case, the underlying transformer of BLIP has 12 layers, each layer has 12 heads. So, we have 144 copies of A in total. Similar to previous works [165], we aggregate the 12 A from the same layer and use RefCOCO+ [205, 165]¹ to perform a grounding study to get the layer that has the best image-text grounding quality. In our case, the best layer is the 7th layer of BLIP. We use this layer’s cross-attention by default, but also allow users to switch to other layers for exploration (through the widget in Figure 4.1-b2). Second, there are also some image resizing operations when computing the Grad-CAM of images, as the images may not be in a square shape. Grad-CAM is resized into the original image size and pixel-wise accumulation is conducted to generate the Grad-CAM for each patch (see our Appendix for these details).

Each column of $M^\cup$ reflects the attention strengths between a single word and all segments. We use these values to color the points in the *Image Segment* view when a word is selected. For example, in Figure 4.1a, the word **fish** is selected, its attentions to all segments are shown in the scatterplot of Figure 4.1b (darker orange: stronger attention). Clicking a point shows the segment, and the spatial distribution of the selected word’s Grad-CAM over the segment will be superimposed as a heatmap (red→blue: strong→weak). For example, in Figure 4.1-b3, the fish head received the most attention from the word **fish**. The lasso-selected segments can also have this overlaid heatmap, as shown in Figure 4.1-b1.

If a segment is among the top- k ($k=3$ by default) segments that received the most attention from a word, we say the segment is *covered* by the word. Iterating through all words, we get a coverage number for the segment, reflecting how many words strongly attend to it. The scatterplot in the *Image Segment* view can also be colored by individual segments’ coverage.

¹RefCOCO+ can be considered as a ground-truth dataset, in which what patches each word should attend to have been well annotated. We have included details of the grounding experiments in our Appendix.

As shown in Figure 4.11a, segments with increasing coverage are mapped to points from yellow to green. The green segments are covered by multiple words and are often the dominant features. For example, the green cluster of fish segments in Figure 4.11b are covered by the words **fish**, **large**, and **swimming**. In contrast, the yellow segments are not sufficiently covered by any caption words, which deserve more attention for caption quality improvement.

Each row of M^U reflects the attention that the corresponding segment has received from all caption words. When a segment is selected, the caption for its residing image will be shown and we use yellow background to highlight the words from which the segment received most of its attentions. For example, in Figure 4.1-b3, the selected segment received attentions mainly from the words **large** and **fish**. The word **fish** is in red, indicating it is the currently selected word from Figure 4.1a.

4.1.4.3 Steering Caption Generation

The LM part of BLIP generates captions based on two inputs (Figure 4.2a), the starting prompt and image patches' embeddings. Therefore, without updating the pre-trained BLIP, we can accomplish **G3** of caption steering by injecting human knowledge into these two inputs.



Figure 4.6: Steering caption generation with guiding prompts.

First, the starting prompt can guide the generation of certain words in the caption. General prompts, such as “a picture of” or an empty string, are often used by default, allowing the model to look at different features equivalently. In cases where we want to generate captions for a specific feature, the prompt can be customized towards that feature. For example, in Figure 4.6,

the old captions generated with the general prompts discard the **sunglasses** (left) and **cars** (right) to compromise for other more conspicuous features. Inserting the verb “**wearing**” and location descriptor “**by the sidewalk**” into the respective prompts (texts in green) leads the model to include these features back in the captions. This is also a proof that the pre-trained BLIP was able to recognize these features beforehand, but considered them less important.



Figure 4.7: Steering caption generation by directly emphasizing (left, weight=5) or de-emphasizing (right, weight=0) the **glasses** patches.

Second, we can also steer the caption generation process by directly emphasizing or de-emphasizing different patches. As shown in Figure 4.2, the input image is decomposed into p^2 patches, each is encoded into a d -dimensional vector through the ViT. To guide the BLIP (LM) model to generate captions for certain visual features, we can force the model to look at them by increasing the corresponding patches’ weight. On the contrary, weights can also be decreased for patches with negligible features. Specifically, we create a p^2 matrix, all elements of which are initialized to 1, and overlay this matrix on top of the image of interest. In Figure 4.7, two examples show the results of emphasizing and de-emphasizing the **sunglasses** feature, respectively. On the left, the caption for the original image does not contain **sunglasses**. After scaling up the weights of sunglasses’ patches (in white squares) from 1 to 5, we get a new caption that includes **dark glasses**. To reverse this process, on the right, the original caption of the selected image involves **sunglasses**. After scaling down the weights to 0 for the corresponding patches, the word **sunglasses** disappears. Instead, the words **black shirt** appear, as the patches for the shirt region receive relatively more attentions shifted from the **sunglasses** pixels.

The *Interactive Caption* view in Figure 4.1c provides an interface to directly apply the above two manipulations. Users can directly type a prompt in Figure 4.1-c2 as the starting prompt. By clicking a pixel of the selected image in Figure 4.1-c1, the patch that the pixel resides in will be selected and marked as white. Users can then increase or decrease the weight of the corresponding patches. After clicking the “Generate” button, a new caption will be generated with the specified prompt (texts in green) or weights applied to the selected patches (Figure 4.1-c3).

There are two reasons why we steer caption generation by manipulating the inputs instead of fine-tuning the model. First, interacting with inputs is more efficient than fine-tuning large ML models. The latter will take significant time and resources. Second, the interaction with inputs is more intuitive, whereas model fine-tuning requires sophisticated data augmentation and may introduce new issues like overfitting.

Note that our caption steering solutions also have certain limitations. First, we assume the captioning model is able to recognize the desired features. Based on this, we guide the model to focus on the desired features and generate descriptions for them. However, if this assumption does not hold, the model can never generate desired captions without further fine-tuning. Second, the patches’ weight in the second approach is an important hyperparameter, but we currently can only get its value through a trial-and-error process and an overlarge weight will disrupt the caption’s semantics. We have investigated several ways to fix this problem, such as normalizing the weight values or smoothing the p^2 weight matrix (with a Gaussian smooth). These optimizations brought some improvements in different cases but no consistent solutions have been established. Our domain experts also found this part particularly intriguing and have initiated some theoretical analyses on it.

4.1.5 Case Study, Findings, and Experts’ Feedback

This section presents studies conducted together with five deep learning experts to show how our system meets the three goals. All experts are full-time research scientists with special interests in transformers, model pre-training, and multi-modal learning. The experts all have 5~10 years of experience in deep learning and have worked on various vision and language models. We studied our system with them following the protocol of *guided explorations + think-aloud discussions* in multiple sessions. Each session extended over a span of 1~2 hours, during which we provided comprehensive explanations on the system components and walked the experts through various cases. The experts were also encouraged to engage with our system, share their insights on our

approach, and offer recommendations for enhancement. At the end, we conducted open-ended interviews with them to collect their feedback (Section 4.1.5.4). Two image datasets were used in the studies: the training set of ImageNet [173] and the validation set of COCO [174]. It should be noted that these datasets have no overlap with the training sets of BLIP or SAM, and our method can be easily adapted to other image datasets, given the strong generalizability of the pre-trained models.

4.1.5.1 Large-Scale Image Dataset Exploration (G1)

ImageNet [173] is often used to train image classifiers. To better differentiate different classes, it is crucial to know the dominant features within individual classes and potential confusion between classes. Therefore, we explore this image dataset class-by-class.

From the header of the *Co-Occurrence Graph* view, we picked the **tench** class, as it has been well studied in early works [148] and some known data issues have been reported, i.e., images of this class often come with a person holding a tench fish. Starting with this class also works as a sanity check to assess our system’s ability to identify previously reported issues. From the visualization in Figure 4.1a, we have several observations. **First**, the word occurred the most in the generated captions is **fish**, instead of **tench**. This is expected as BLIP is not able to differentiate different types of fish. **Second**, from the nodes’ sizes and links’ widths, we can easily identify the dominant and co-occurred features, i.e., **man**, **hand**, **fish**, and **holding**. This effectively highlights the potential issue of the images (i.e., undesired feature co-occurrence). People should be aware that ML models trained on these images will heavily rely on the presence of **man** and **hand** to recognize an image as a **tench** image. **Third**, Figure 4.1-a1 also reflects that **man** occurs much more frequently than **woman** and **person**, suggesting a gender skew in fishing activities. Consequently, models trained on these images may involve potential gender bias as well. **Lastly**, exploring different words in the *Co-Occurrence Graph*, we can see the corresponding visual features are very well clustered in the *Image Segment* view. For example, the dark orange cluster in Figure 4.1b corresponds to segments strongly associated with the currently selected word, i.e., **fish**. Similarly, in Figure 4.8, the three dark orange clusters show the segments that the word **man** strongly attends to, i.e., **head+body**, **face+hand**, and **face**. The heatmaps over individual segments further enrich our understanding. The coordinated explorations validate the association between different caption words and image features, enhancing users’ comprehension.

Next, we switched to the **white_wolf** class. In [168], the authors intentionally curated a small

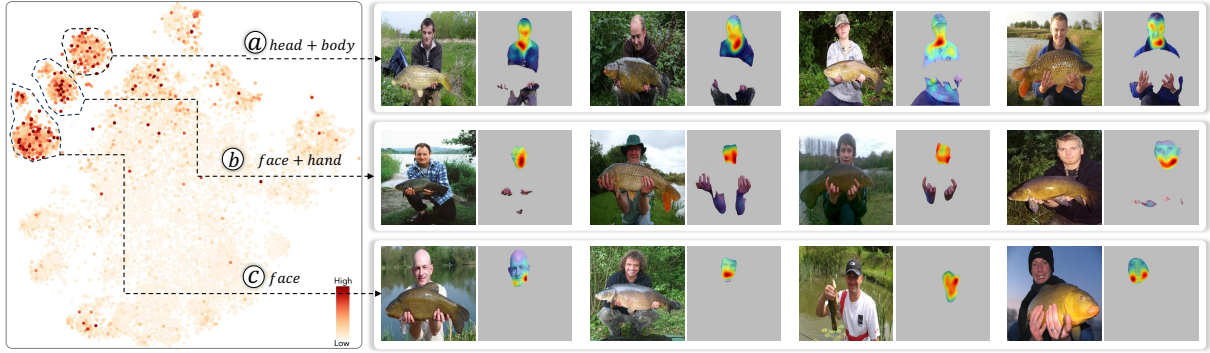


Figure 4.8: For **tench** images, **man** is a dominant feature. The three dark orange clusters show three groups of segments that the word **man** strongly attends to. Some randomly selected segments are visualized on right. The overlaid heatmaps help to disclose the spatial distribution of the attentions.

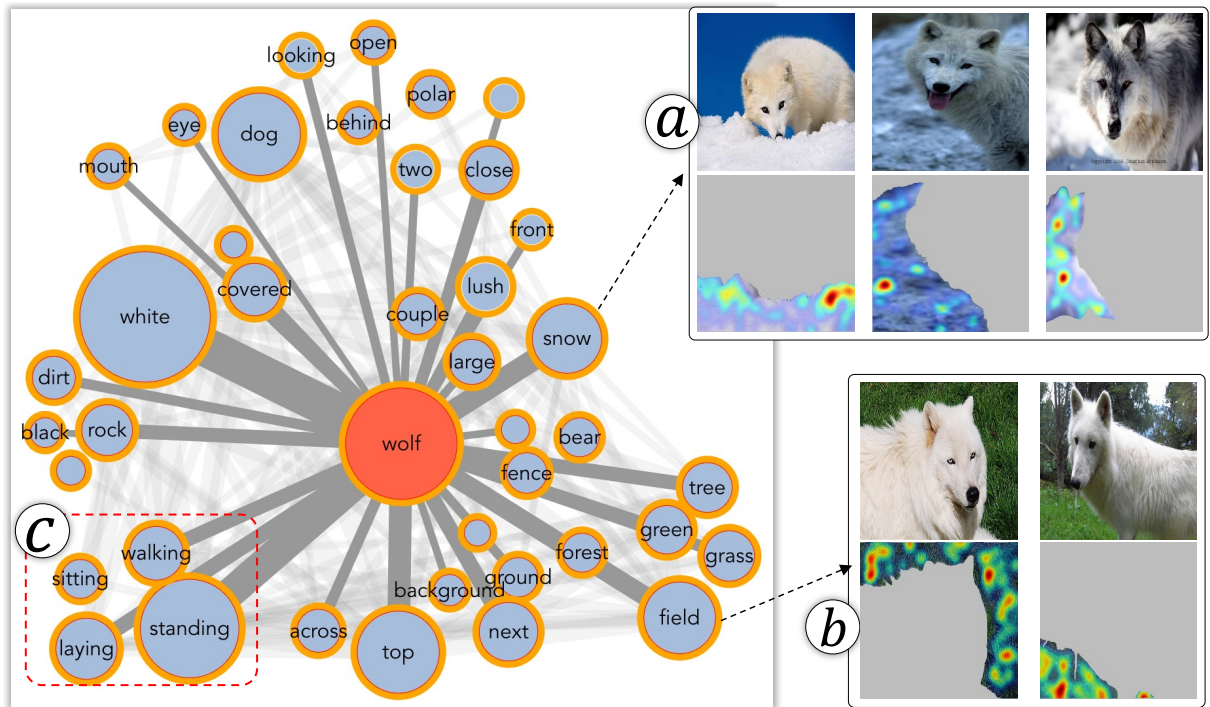


Figure 4.9: **Snow** and **field** frequently co-occurred with **white_wolf**.

number of **wolf** images with snowy backgrounds to explain how undesired feature co-occurrence could lead to a biased model. Inspired by their work, we want to see how severe the co-occurrence between **white_wolf** and **snow** is in ImageNet. As shown by the node-link diagram in Figure 4.9, **snow** indeed frequently occurred in **white_wolf** images, reflected by the corresponding node's size. Clicking on the node of **snow**, we can see a clear cluster of snow segments in the *Image Segment* view (not shown here) responding strongly to the word. Figure 4.9a shows some example segments and the corresponding images. Apart from **snow**, our system also detects other features

that are frequently present in the natural habitat of **white.wolf**. For example, the node for **field** is also large. As shown in Figure 4.9b, this node, along with several other nearby nodes (e.g., **green** and **grass**) discloses other high-frequent background features in **white.wolf** images. Furthermore, the dominant words from the captions also capture the main poses of wolves in the images (Figure 4.9c), such as **standing** and **walking**. The overview from our system provides a holistic picture for the dominant postures of wolves and their living environment.

The *Co-Occurrence Graph* also helps to structurally explore different image features and disclose their imbalanced distributions. We use the COCO dataset [174] to demonstrate this. COCO contains images with annotated objects in 80 categories, and each image involves multiple objects. While exploring the images with both **person** and **tie** objects, we found the word **man** is a high-frequent feature (i.e., the biggest node in Figure 4.10a) whereas **woman** has a much lower frequency. Examining the co-occurrence features of **woman**, we found it has a thick link to **man** but a very thin link to **tie**. This reveals two insights: (1) men occur much more than women in the **person-tie** images, and (2) the occurrence of women is more often accompanied by men than by wearing ties. Examining segments that strongly respond to **woman** and checking the corresponding images in the *Image Segment* view, we verified these insights (Figure 4.10b). Exploring the objects within images, we reveal the imbalanced distribution of genders and their co-occurrence with **tie**. Such an imbalanced distribution might lead to a trained ML model failing to detect ties worn by women, and thus deserves more attention.

More exploration results and discovered insights have been included in our Appendix. The knowledge on the frequency of different image features and their co-occurrence provides valuable insights into a dataset, helping to comprehend the dataset before using it for any model training/evaluation². Although whether a feature co-occurrence should be considered as a data bias or not is very application-specific, it is crucial to reveal it to model designers to support their decision-making. The awareness of this information also helps them select datasets and remain vigilant regarding model issues stemming from the data [175, 176].

4.1.5.2 Caption Coverage and Model Behavior Pattern (G2)

While the image captioning model exhibits remarkable performance, it is not without imperfections. Revealing what common image features that the model often overlooks can shed light

²Note that our focus in this work is data-level exploration, and thus, there is not any downstream ML model trained or evaluated on the explored datasets.

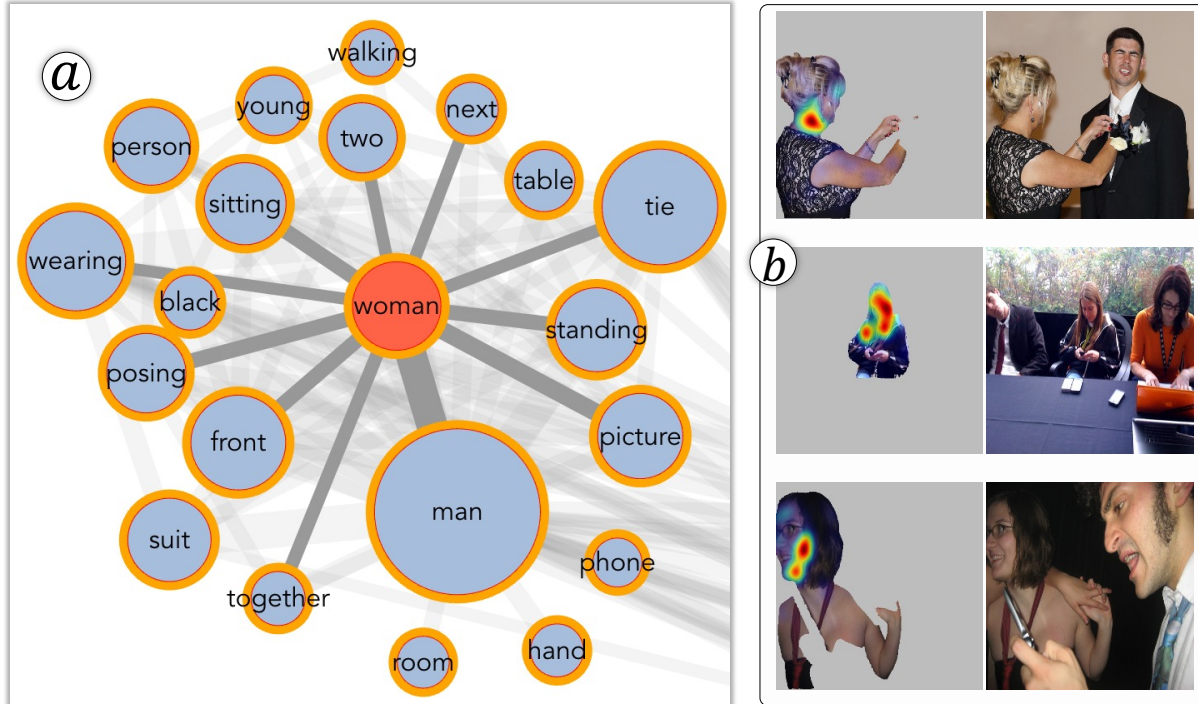


Figure 4.10: Exploring COCO images with both **person** and **tie** objects. on its weaknesses and offer valuable insights for its enhancement. To demonstrate our system’s capability in this aspect, we continue our class-level explorations, as images of the same class often share many common features.

Continuing the **tench** case, we feed all SAM extracted segments from the **tench** images into ViT and use t-SNE to lay out their resulting embeddings in the *Image Segment* view (Figure 4.11a). From the layout, we observed that the segments are well clustered. By individually exploring the clusters through lasso selections, we could easily figure out the semantics of each cluster. For example, Figure 4.11b, 4.11c, and 4.11d show the segments of **fish**, **human face**, and **tree branch**, respectively.

The segments in Figure 4.11a are colored using their word coverage, i.e., the number of unique words strongly attending to the corresponding segments. The color from yellow to green reflects the increasing coverage (see the legend). Evidently, the segments related to **fish** (the two clusters on the right) and **man** (the three clusters on the top-left) received the most coverage. Apart from them, many points/segments are in yellow (i.e., zero word coverage). If the yellow points are dispersed, such as the bottom right region of Figure 4.11a, the corresponding segments have fewer features in common (i.e., fragmented pieces). However, if the yellow points form into isolated clusters, it usually indicates that some common features are inadequately covered by

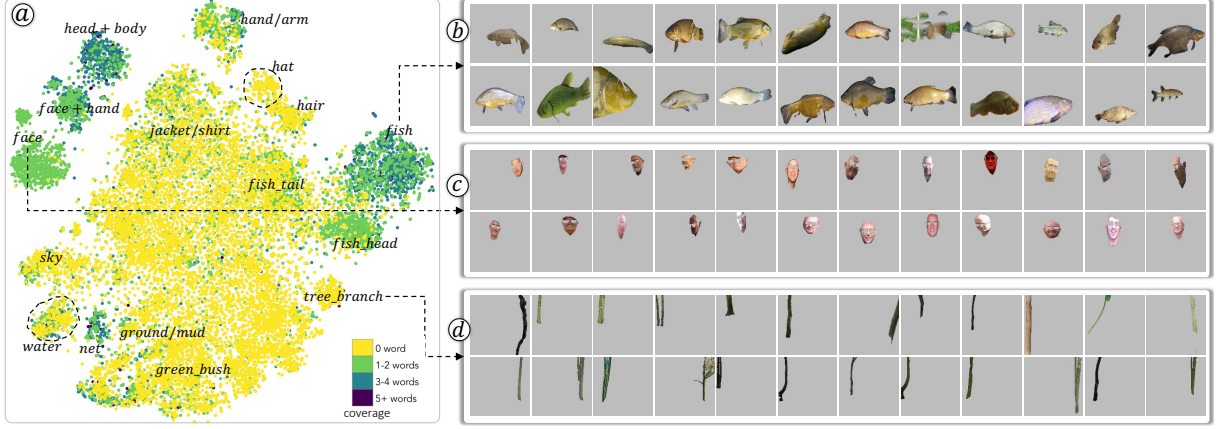


Figure 4.11: (a) The word coverage over segments of **tench** images (color reflects the word coverage, see the legend). Yellow points in isolated clusters are common features that are often overlooked and deserve more attention.

(b, c, d) Three example clusters are explored through lasso selections.

the captions, such as the clusters for **hat**, **tree.branch**, **water** and **sky** in Figure 4.11a.

To verify this, we picked the **hat** cluster for detailed investigations. Figure 4.13a (the middle column) shows some randomly picked **hat** segments. Their residing images are shown in the left column. The generated captions (below the images) do not contain **hat** descriptions. Consistent phenomena were observed in most images with this feature. This recurring behavior pattern, i.e., the model tends to overlook the **hat**, reveals a weakness of the model. Similarly, the segments of background **water** (the bottom-left cluster in Figure 4.11a) are also insufficiently covered, and some randomly picked image examples are shown in Figure 4.13b. Both **hat** and **water** exist in a substantial number of images and thus should present strong occurrences among **tench** images. Nevertheless, with limited control over the caption generation process, the BLIP model struggles to generate descriptions for these features once it has decided to overlook them. Consequently, these features are under-represented in the *Co-Occurrence Graph* view, which compromises the accurate exploration of the image contents (**G1**).

Similar caption coverage insufficiency has also been found in the COCO dataset. In this case, we caption images that contain both **car** and **person** objects. Although the **car** object is known to exist in all images, the generated captions cannot cover it very well, as reflected by the yellow cluster of **car** segments from the *Image Segment* view (not shown here for space limit). As some randomly picked examples shown in Figure 4.13c, this omission is due to the **car** being in the background or outsized by other objects. Although the **cars** do not play key roles in these images, they (along with **person**) are the focus of captioning in this case, and

users would expect to see them in the generated captions.

4.1.5.3 Steering Caption Generation (G3)

With the identified overlooked features, users can decide if they want to update the captions to include them or not. If yes, they can steer the caption generation using the *Interactive Caption* view in two ways.

First, users can update the starting prompt to lead the model toward the desired features. As shown in Figure 4.1c, a **tench** image containing the **hat** segment in the yellow cluster of Figure 4.11a is selected. The original caption, generated with the prompt “a picture of”, does not contain **hat**. However, after modifying the prompt to “the person is wearing”, we successfully guided the model to generate a caption with the word **hat**. The new prompt can also be applied to all images containing **hat** segments for batch processing. For example, in a lasso selection of 214 images containing segments in the **hat** cluster of Figure 4.11a, 193 newly generated captions contain the word **hat/beanie/hoodie**, resulting in a success rate of 90.19% (193/214). For the 21 failed cases, the words **jacket**, **gloves**, and **goggles** primarily respond to **wearing** in our prompt. For these cases, we have to explicitly emphasize the **hat** pixels to better steer the model (explained next). Also, 9 out of the 21 images indeed have no **hat** features, but the SAM extracted segment is very similar to a **hat**. We have included the full details of this batch processing experiment in our Appendix.

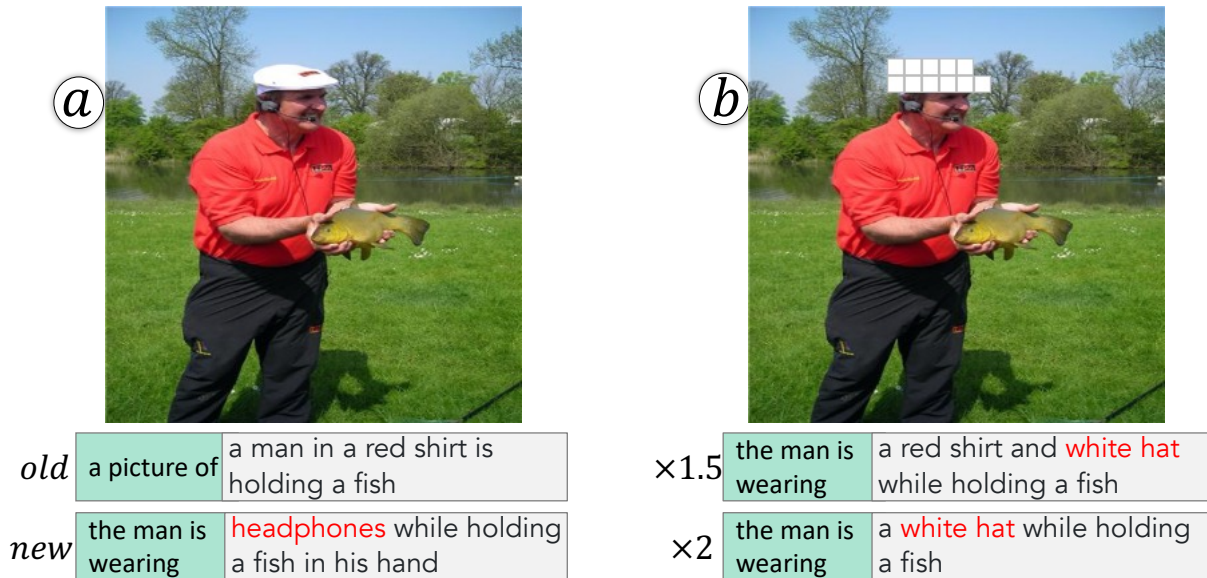


Figure 4.12: Iteratively steering the model to generate descriptions for **hat**.

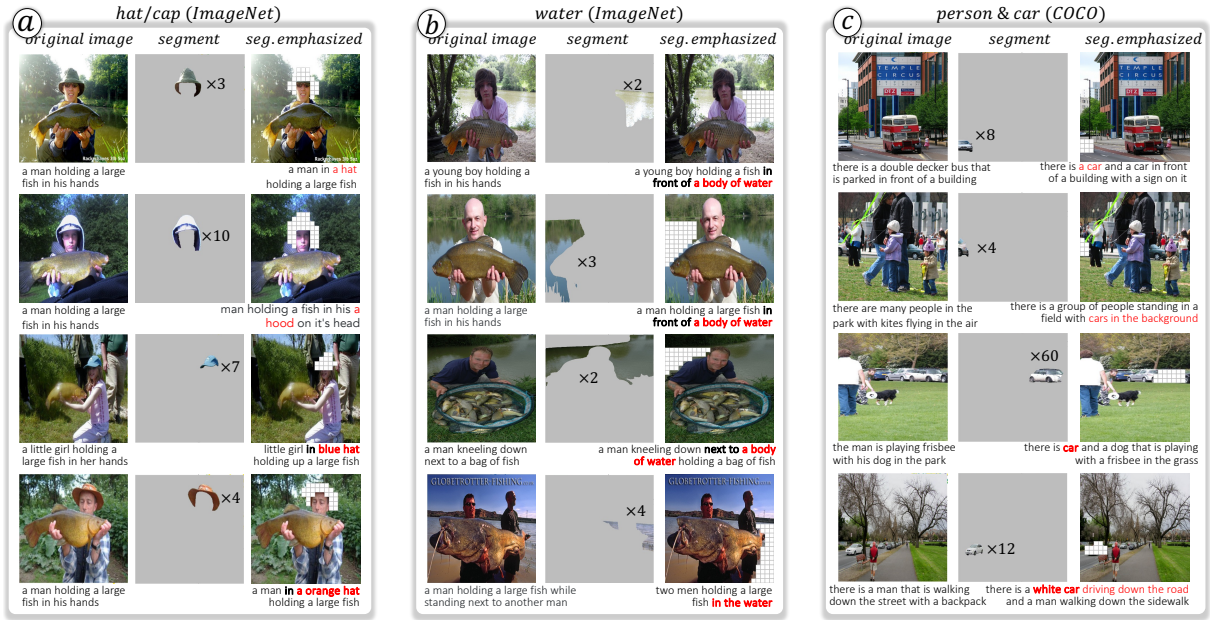


Figure 4.13: ImageNet **tench** class images with the (a) **hat** and (b) **water** features, and (c) the COCO **person+car** images. The images are randomly picked from the clusters of segments with the respective features. The images’ original captions and updated captions are shown below the images.

Second, users can more explicitly specify what image features the model should look at. In Figure 4.12a, after updating the prompt to “the man is **wearing**”, the word **headphones** appears, responding to **wearing** in our prompt. However, the feature of interest is the **hat**. Therefore, we select the **hat** patches and increase their weight to 1.5. From the result shown in Figure 4.12b, we can see **white hat** successfully appear in the new caption. Further increasing the weight to 2 will make the model even more focus on the **white hat** region and ignore the **red shirt**. Note that the weight value for different images could be different and we found the proper value through trial-and-error explorations. Our interface instantly generates new captions while users emphasize/de-emphasize different patches, providing immediate feedback to accelerate the explorations. A general heuristic for weight tuning is to keep increasing its value until the desired word appears. However, an overlarge weight can also break the semantics of the caption. In these cases, we may fail to generate captions with desired words, as the model cannot recognize the corresponding features.

Figure 4.13 presents more randomly selected caption steering examples for (a) **hat**, (b) **water**, and (c) **car** features, respectively. For each set of examples, the left, middle, and right columns show the original images and captions, the emphasized segments and applied weights,

the updated images and new captions, respectively. In most cases, the weight is between 2 and 10. However, there are also cases where we need to use a very large weight, e.g., 60 in the third example of Figure 4.13c.

4.1.5.4 Domain Experts’ Feedback

After walking the five experts ($E_1 \sim E_5$) through various cases and ensuring that they have a comprehensive understanding of our solution, we conducted open-ended interviews to gather feedback from them.

In general, all experts expressed positive views on our idea and the developed system. For image exploration (**G1**), E_1 liked the function of flexibly pinning individual nodes through drag-and-drop, which is very helpful to him when navigating the dynamic node-link diagram. While exploring the case in Figure 4.9, he noticed that there is a big node for **dog** and it is not connected to **wolf**, indicating **dog** and **wolf** never appeared in the same caption. This reflects that the model either correctly recognized the animal as a **wolf** or misclassified it as a **dog**. He regarded this as an inherent limitation of BLIP in differentiating wolves from dogs. E_4 was particularly intrigued by the idea of using automatically generated captions to succinctly summarize large-scale image datasets. He has worked on de-biasing pre-trained word embeddings for years. One recurring challenge he faces is quickly confirming the existence of potential biases. Our approach could serve as a valuable starting point before applying his de-biasing algorithms. E_3 and E_5 expressed their appreciation for the seamless coordination between the node-link diagram and scatterplot. They further brainstormed using them to hierarchically explore image features. For example, in Figure 4.11a, the **head+body**, **face+hand**, and **face** can be considered as clusters nested under the **man** cluster. Such a hierarchy can significantly expedite explorations and provide deeper insights into image features.

For caption coverage (**G2**) and steering (**G3**), E_2 successfully generated the word **hat** in Figure 4.1c by editing the prompt. Instead of stopping there, he wondered what would happen if he further enhanced the **hat** patches. He selected those patches, set their weight to 1.5, and generated the caption again. Surprisingly, the new caption replaced **hat** with **beanie** (a hat without brims), which is a more accurate description. Both E_2 and E_5 were impressed by the associations encoded in cross-attention and appreciated the visualizations that intuitively expose them. E_5 liked our idea of computing the coverage for each segment and commented that the segments covered by many words may also be problematic, as the received attention is too diverse. He

suggested using some metrics to quantify the diversity level of those words.

On the other hand, the experts also mentioned some imperfections and desired features. For example, E_5 wished the weight adjustments in caption steering could be more automatic and applied to batches of images to minimize human effort. E_1 and E_2 brainstormed using SAM to assist in selecting patches with desired features, i.e., by clicking on any pixels within the segment to be captioned and having SAM automatically select the related patches. E_5 also suggested using large language models to help the prompt-engineering process and generate words that better provoke the model to generate desired words. These suggestions provide promising directions to further extend our work.

4.1.6 Discussion, Limitations, and Future Work

Hyper-parameters. During our data pre-processing, several parameters are used to filter out less meaningful data. Specifically, when processing captions, we filter out words that rarely appear, as the analysis should focus on the most prevalent features. When processing images, segments that are too small or with very similar features are filtered out to avoid meaningless or repeated features. The values of the filtering thresholds can be easily adjusted and these parameters do not directly impact the effectiveness of our approach, as the eliminated words/segments always represent less dominant features.

Pre-trained models’ quality. Our exploration of image datasets heavily relies on pre-trained language-image models. Consequently, quality issues stemming from these models can impact our exploration. While the pre-trained BLIP can identify distinct objects in each image, it is limited in differentiating subcategories and finer-granularity features. For example, when exploring the `white_wolf` images in Figure 4.9, there is a big node for `dog`, indicating the BLIP model occasionally misclassified `wolf` as `dog` and generated inaccurate captions. However, as the availability of high-quality pre-trained models continues to grow, we anticipate that this problem will become less severe in the future. Additionally, although we focus primarily on BLIP in this work, our system can be effortlessly applied to other image captioning models, as long as they use cross-attention for image-text association.

The node-link diagram. Having explored multiple image datasets with our system, we gradually noticed two insufficiencies in our *Co-Occurrence Graph* and are working on some remedies. First, despite the force-directed layout, we are considering incorporating word similarity into the nodes’ layout. For example, the words `green`, `grass`, and `tree` in Figure 4.9b should be

automatically positioned closer to help the exploration (now they are manually pinned together). The similarity between words can be measured by the distance between their word2vec embeddings. This will be particularly valuable when navigating larger node-link diagrams. Second, we are also considering implementing a node filter based on the words’ part-of-speech. For example, the word **large** in Figure 4.1a and **white** in Figure 4.9 are adjectives describing the respective objects. Their substantial size may divert users’ attention from analyzing the co-occurred objects, which are typically nouns. Existing packages, like NLTK [201], can be used for this filtering.

Caption steering. We have also identified two limitations in our caption steering approach. First, we observed that the generated captions can be very sensitive to the prompt. For example, in Figure 4.12a, the word **headphones** would disappear if changing the prompt from “**the man is wearing**” to “**the person is wearing**”. Second, we currently can only derive the proper weight for image patches through a trial-and-error process. The consequences of both limitations require users to iteratively play with our system to generate desired captions. On a positive note, the instantly generated captions offer valuable feedback to help users adjust the prompt/weight, which mitigates the limitations to some extent. In the future, we plan to work with domain experts toward more automatic prompt engineering and weight searching algorithms.

4.1.7 Conclusion

In this work, we introduce a visual analytics system to explore large-scale image datasets through the captioning power of a pre-trained language-image model, i.e., BLIP. Our system is also capable of evaluating the generated captions by revealing their coverage over different image features. Focusing on the insufficiently covered features, we provide an interface to steer the caption generation process. Through multiple cases conducted on two large-scale image datasets with multiple domain experts, we validated the effectiveness of our system.

Chapter 5

Conclusion

As machine learning (ML) relentlessly surges forward, pushing frontier technologies across diverse scientific domains, a deeper truth emerges that genuine progress demands not only advanced algorithms but also a clear commitment to transparency and understanding. Compared to the rapid advancement of ML architectures, the development and application of interpretability techniques remain relatively underexplored. This dissertation first develops interpretability techniques and then investigates their practical application through VA across three essential phases of ML development: model understanding, evaluation, and improvement.

First, I develop VA and ML interpretability solutions to enhance model understanding across diverse applications. For the understanding of water distribution system optimization (Section 2.1), I visualize the interpretable decision tree to characterize the ensemble simulation data distribution and optimization rules. For the understanding of vision transformers (Section 2.2), I focus on using VA to assess attention head importance, reveal spatial attention patterns, and summarize common attention behaviors. These solutions advance interpretability in ML by providing actionable and visually accessible insights into black-box models, and lay a solid foundation for next-step model evaluation and improvement.

Second, I integrate interpretability methods into interactive visualizations to evaluate model reliability and robustness. To evaluate the reliability of prediction models on medical data (Section 3.1), I utilize VA to examine the consistency of SHAP values in relation to feature value changes. To assess robustness against adversarial attacks (Section 3.2), I rank neurons based on their receptive fields to explain their vulnerability. Through both instance-level and group-level interactive analysis, these visualizations enable comparisons of model reliability and help diagnose vulnerabilities within the model’s internal neurons.

Finally, I introduce interpretability-driven enhancements within interactive visualizations to improve model comprehension and performance. To understand and improve vision-language models' generation capability (Section 4.1), I analyze cross-attention mechanisms to reveal connections between generated captions and specific image features, highlighting potential biases. To address insufficient coverage of certain image features, I develop a visual interface that allows researchers to dynamically adjust the captioning process, mitigating identified weaknesses. By combining visual and textual analysis, this system enables targeted improvements, validated through case studies to enhance both model generation and user-centered understanding.

In conclusion, this dissertation advances the field of ML research by developing and applying VA and ML interpretability methods to address key phases of model understanding, evaluation, and improvement. By prioritizing transparency, reliability, and user-centered design, this work lays a foundation for future research that strives not only for model accuracy but also for accountability and adaptability in real-world applications. As ML continues to evolve, the frameworks and methods established here underscore the essential role of interpretability in ensuring that these powerful technologies remain comprehensible and trustworthy.

Appendix A

Appendix for Section 2.2

A.1 Results From Other ViT Models

To prove the system’s generalizability and verify the findings’ consistency, the explorations are repeated on another three ViTs (Table A.1). Most of the findings across the four models

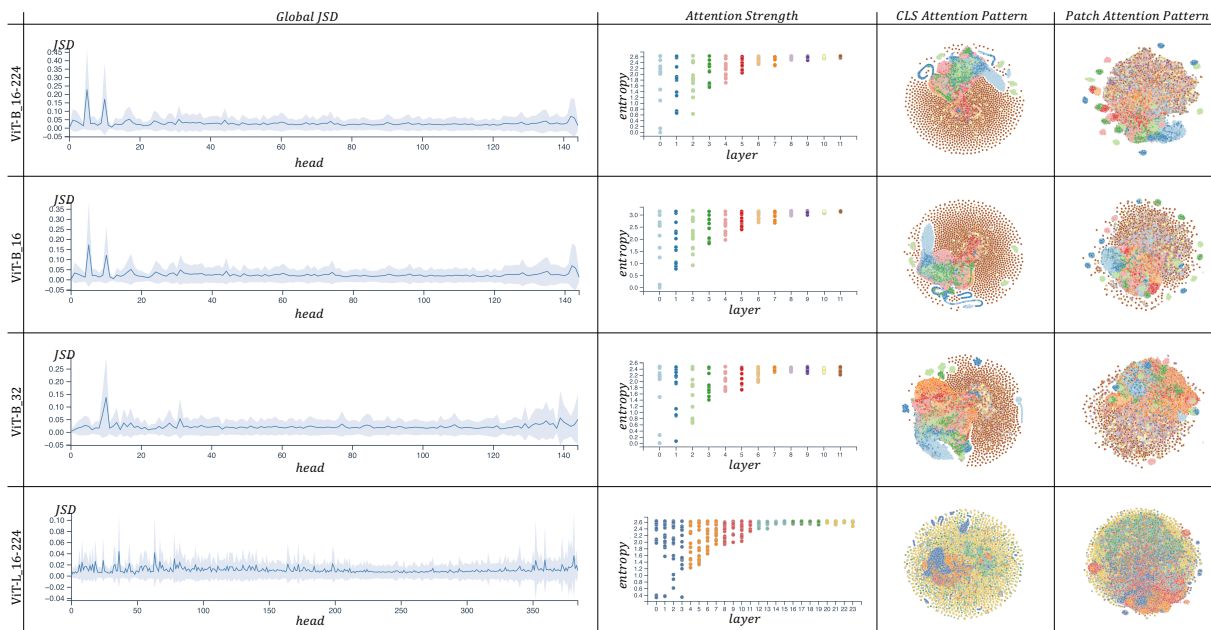


Figure A.1: Comparison of visualizations for four different ViTs in Table A.1. Each row presents the overall JSDs, entropies of attention strengths, and the dimensionality reduction results of CLS attention patterns and patch attention patterns. For the ViT-L_16-224 model, we encode every four layers with one color (since it has more layers than the other three). Consistency can be observed in the evolution of entropies through layers, and the trend of attention patterns from lower to higher layers. Different from the first three models, there’s much more oscillation in the JSDs of the ViT-L_16-224 model, and there’s no globally important head, probably because there are many more heads in this model, so that the functionalities of each head become less orthogonal with others.

ViT Models	Width (w)	#Layer (l)	#Head (n)	#Patch (p)	Patch size (pz)
ViT-B.16-224	224	12	12	14	16
ViT-B.16	384	12	12	24	16
ViT-B.32	384	12	12	12	32
ViT-L.16-224	224	24	16	14	16

Table A.1: Four pre-trained ViT models from [76] are compared in this Appendix. All can be downloaded from the Google Cloud Storage (download [link](#)). ViT-B.16-224 is the one we used in Section 2.2. The ‘B’ and ‘L’ in the model names mean ‘Base’ and ‘Large’.

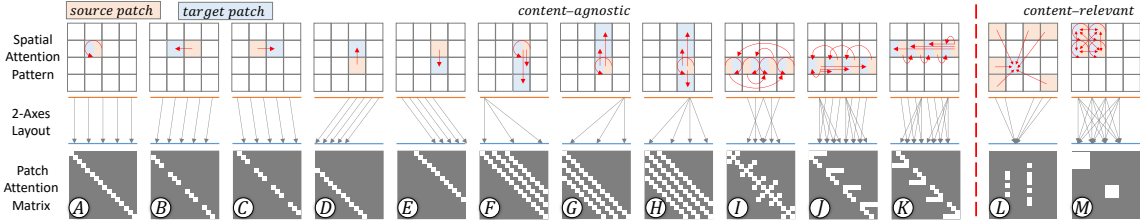


Figure A.2: Illustrative figure for the 13 attention patterns mentioned in Section 2.2.5.3, i.e., Figure 2.15.

are consistent. In Figure A.1, we compare the JSDs, the entropies of attention strengths, the dimensionality reduction results of CLS attention patterns and patch attention patterns for the four models. The similarities and differences across the four ViTs are summarized as follows.

Similarities. *First*, the reasons for a head to be important are similar, e.g., strong self-attention of the CLS/patch tokens or strong attentions to the class-related contents between the CLS and patch tokens. *Second*, the attention strength distributions inside individual heads across layers are similar in all four ViTs, i.e., lower-layers include heads with both small and large entropy, but higher-layers heads all show a large entropy. *Third*, the three new ViTs also present the 13 head attention patterns (except patterns J and K in ViT-B.32 model, which is discussed in Section A.2.1) and their occurrences across layers are also similar, i.e., the two content-agnostic patterns (*diagonal* and *horizontal*) are often observed from lower attention layers; and the two content-relevant patterns (*vertical* and *block*) are often observed from higher layers.

Differences. *First*, the occurrences of important heads from lower layers are different. The important heads from ViT-B.16 are very similar to those of ViT-B.16-224. However, for ViT-B.32, there is only one globally important head (layer 0, head 9), the third row of Figure A.1. For ViT-L.16-224, no head shows global importance, and we observe more diversity in each head’s local importance to individual images. This might be caused by the much larger number of heads in this model (i.e., it has 24×16 heads, which is much more than 12×12 in the other three). The

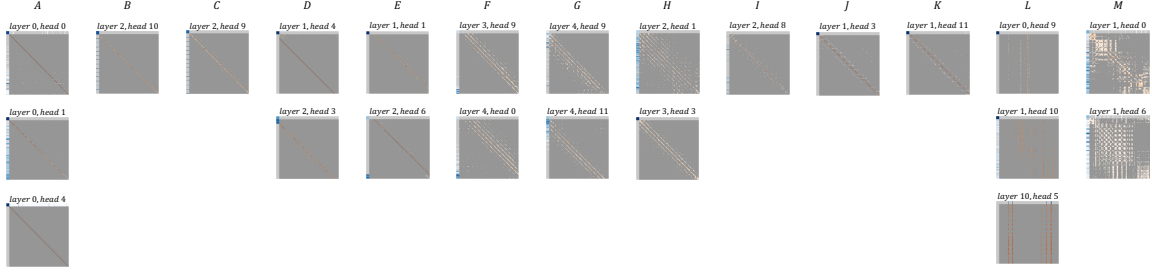


Figure A.3: Example heads in the ViT-B_16-224 model that contain the summarized patterns in Figure A.2. Attention matrix size: $((14 \times 14 + 1) \times 14 \times 14 + 1)$.

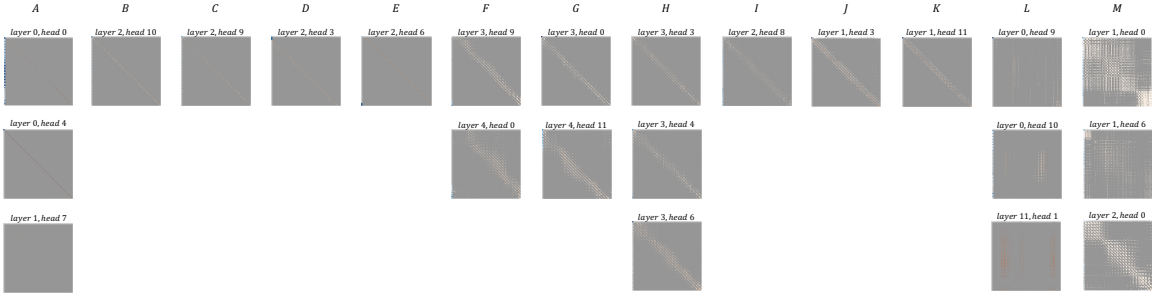


Figure A.4: Example heads in the ViT-B_16 model that contain the summarized patterns in Figure A.2. Attention matrix size: $((24 \times 24 + 1) \times 24 \times 24 + 1)$.

larger number of heads may result in less orthogonal functionality distribution across heads. As a result, pruning one head at a time may not cause a significant change in the outcome, as other heads will easily compensate for its role. *Second*, although the 13 patterns can also be found in the other three ViTs, they appear in different heads of ViT-B_32 and ViT-L_16-224 due to the randomness in learning. Also, for ViT-L_16-224, we found more heads with mixed patterns and fewer isolated clusters in the last row of Figure A.10, which we believe is also a result of its much larger number of heads.

A.2 More Details on Head Attention Patterns

In this section, we first present example heads (from different ViTs) that contain the 13 patch attention patterns summarized in Figure A.2, i.e., Figure 2.15 of Section 2.2.5.3. Then, we discuss several exceptional patterns which can potentially be added to the pattern list.

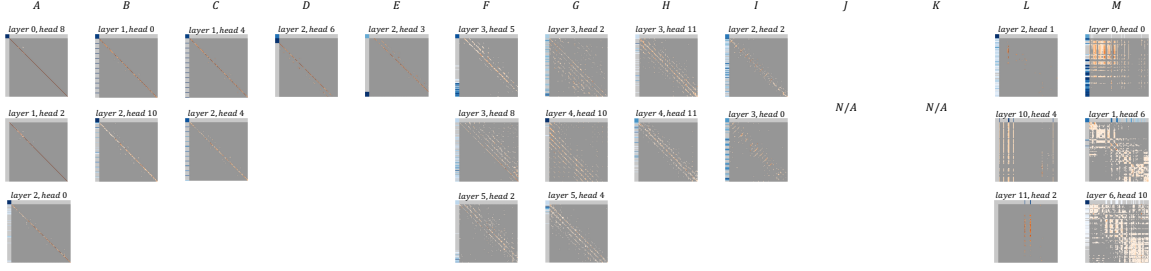


Figure A.5: Example heads in the ViT-B_32 model that contain the summarized patterns in Figure A.2. This model lacks patterns J and K, as there might be fewer possibilities of attention directions for a fewer number of tokens. Attention matrix size: $((12 \times 12 + 1) \times 12 \times 12 + 1)$.

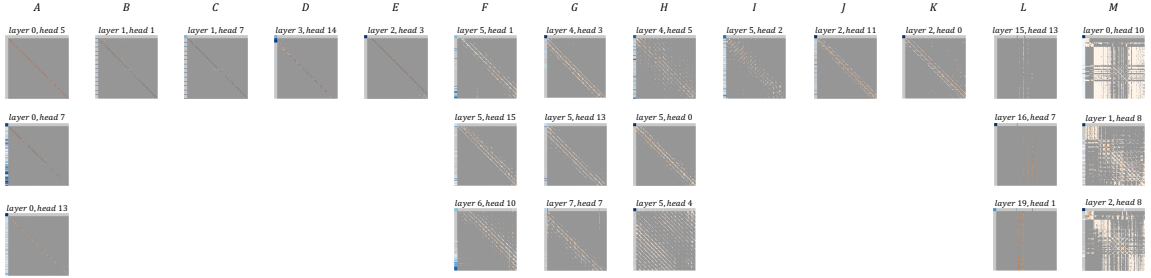


Figure A.6: Example heads in the ViT-L_16-224 model that contain the summarized patterns in Figure A.2. Attention matrix size: $((14 \times 14 + 1) \times 14 \times 14 + 1)$.

A.2.1 Pattern Examples

Figure A.3, Figure A.4, Figure A.5, and Figure A.6 present the real attention patterns of different heads in the ViT-B_16-224, ViT-B_16, ViT-B_32, and ViT-L_16-224 model, respectively (see Table A.1 for the details of these four ViTs). We stack the attention maps vertically for each pattern and include the head IDs above each map. The letters on the top of each column indicate the pattern ID in Figure A.2. Most patterns are identified among the heads of the four models, except patterns J and K for ViT-B_32 (Figure A.5). This may be caused by the smaller number of patches in the model, as there will be much fewer alternatives of attention directions.

The varying number of heads across different patterns indicates diverse types of attention patterns included in our summary. On the one hand, some patterns can be identified in multiple heads, indicating that these heads share the same functionalities. For example, pattern A, the main diagonal pattern, is found in three example heads for each model, which means patches attend the most to themselves for all of these heads. On the other hand, some patterns are identified in only one head but are consistent across different images. These patterns represent functionalities unique to a specific head. For example, pattern I, the counter-diagonal pattern, is only found in layer 2 head 8 for the ViT-B_16-224 model, which means that the functionality is

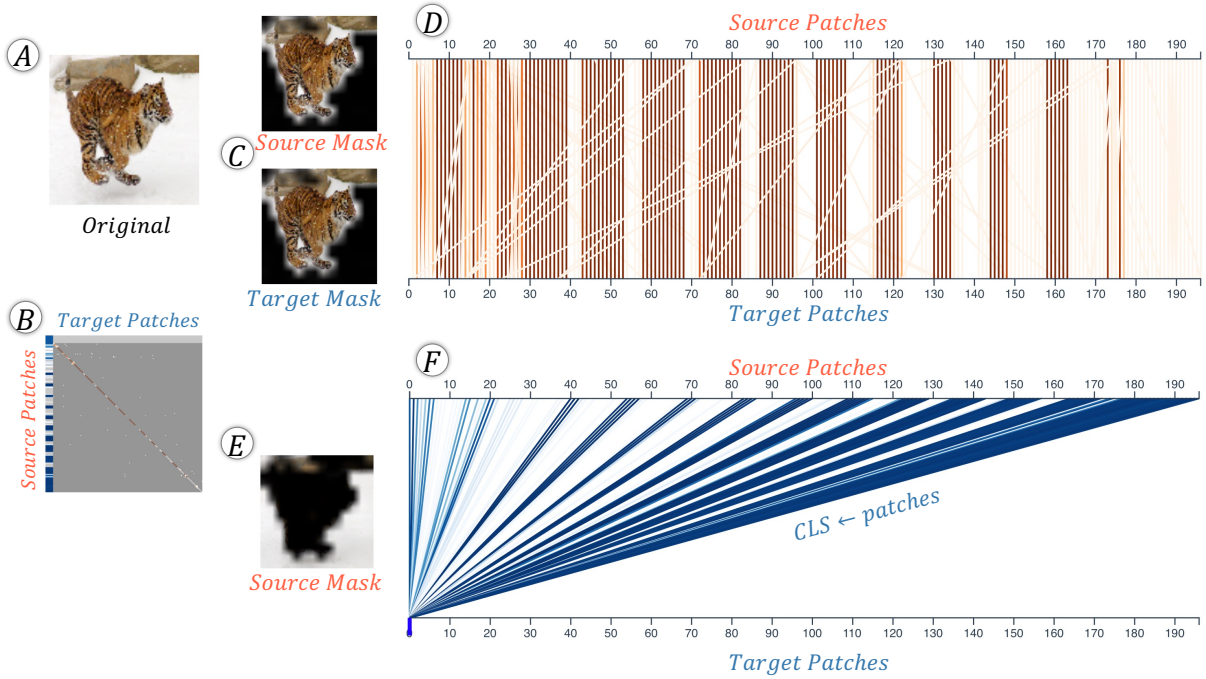


Figure A.7: Example content-relevant head with the diagonal pattern. For the original **tiger** image, only the patches related to the **tiger** are highlighted (C), which means these patches assign higher weights to themselves than the background patches (D). In contrast, background patches with lower self-attentions will assign higher weights to the CLS token (F), resulting in a flipped effect of the image mask (E).

unique to this head (left and right patches symmetrically attending to each other in each row).

A.2.2 Special Attention Patterns

1. Content-Relevant Diagonal Patterns In most cases, the diagonal pattern is content-agnostic. However, we also found few exceptions where the diagonal pattern is relevant to the content. *Layer 0, head 0* from ViT-B₁₆-224 is such an example. It is observed from Figure A.7B that this head has the dominant diagonal line pattern, which means that every patch assigns the largest attention to itself. However, in Figure A.7C, the main object (**tiger**) in both the source and target masked image is highlighted. This is caused by the variation of self-attention strengths across patches. As presented in Figure A.7D, patches containing pixels of the **tiger** assign larger attention weights to themselves, and the background patches assign lower attention weights. Meanwhile, patches with low attention weights to themselves assign high weights to the CLS token (Figure A.7F), which results in a “flipped” CLS attention mask (Figure A.7E), in contrast to the mask of patch attentions. Because of the “segmentation” effect of the attention weight variation, this head can also be considered content-relevant.

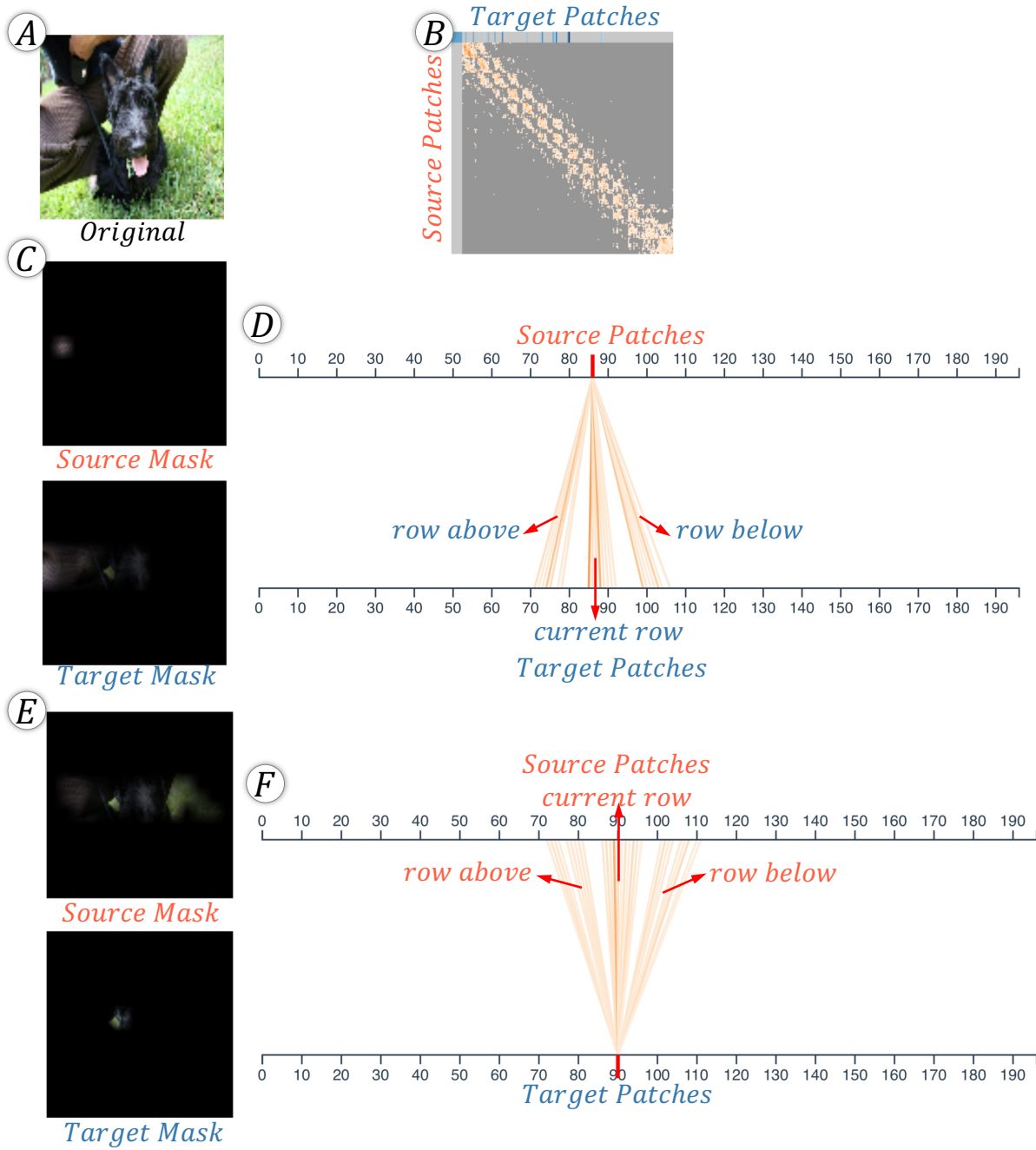


Figure A.8: An example of the content-agnostic block pattern. *Layer 3, head 2* from ViT-B.16-224 exhibits such a pattern across different images. The heatmap of the attention matrix is shown in B. This pattern represents that each row in the image is looking at one row above, the row itself, and one row below. The image masks and the two axes in C, D, E, and F describe this pattern through an example source patch and an example target patch.

2. Content-agnostic Block Pattern Although most heads with block attention patterns are content-relevant as the blocks indicate important image regions, there are minor cases where the block patterns and their locations are consistent across images and thus are agnostic to the content.

For example, in Figure A.8, we show a head in the ViT-B_16-224 model (layer 3, head 2) that consistently exhibits block attention patterns along the diagonal line. The heatmap of attention matrix in Figure A.8B depicts the pattern in detail, where there is a string of small blocks along the main diagonal line and two strings of blocks along the two sub-diagonal lines. This pattern can be viewed as a combination of the multi-diagonal pattern (pattern H in Figure A.2) and the block pattern (pattern M in Figure A.2). In the multi-diagonal lines pattern, each patch attends to itself and one patch above and below. So for the current pattern of multi-diagonal blocks, each patch is expanded into each row of patches in the image, and each row attends to itself, one row above and below, where the attentions are between all patches in the corresponding rows. In Figure A.8C and D, one example source patch is hovered, and it is clear that there are three main sections of attentions from this patch: the ones to the row above, the ones to the current row, and the ones to the row below. Figure A.8E and F show a similar example where one target patch is hovered, and the attentions to this patch can also be split into three groups of the attentions from the row above, the current row, and the row below.

3. Heads with Directional Patterns Figure 2.16 of Section 2.2.5.3 presents one head where all patches attend to the patch below them and the last row of patches attend to the CLS token. In fact this attention strategy also appears in the other three heads with the directional patterns from layer 2 (i.e., Figure 2.14G-J of Section 2.2.5.3). In Figure A.9, we present the attentions between patches and the attentions related to CLS separately for the four heads that make patches attend to their top, bottom, right, and left patches. In addition, we illustrate the overall attention directions in the last column of Figure A.9.

A.3 Image Context Visualization

A gray-scale transparent mask is laid over the original image to show the source and target attentions in the system (i.e., Figure 2.12D of Section 2.2.5.3). Apart from this method, using a

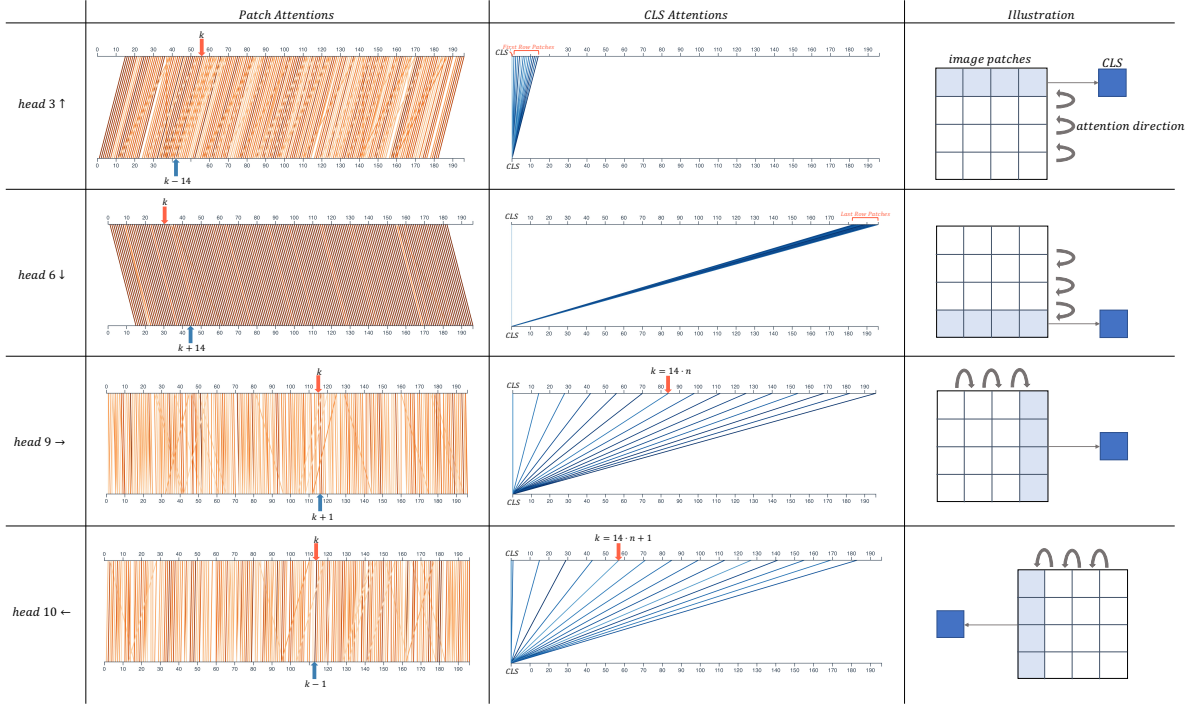


Figure A.9: Example heads with the four directional attention patterns (up, down, right, left). In ViT-B_16-224 (14×14 patches per image), these four heads are all from layer 2 (i.e., Figure 2.14G-J in Section 2.2.5.3). The top/bottom axis represents the sequence of source/target tokens in the first two columns of this table (orange and blue colors are for patch and CLS attentions). The last column illustrates the attention pattern in the image space.

transparent heatmap over the image to visualize the attention is also very common (colors from red to blue indicate values from large to small). In Figure A.10, we present several examples of relevant semantics captured by the attentions through both the gray-scale masks and heatmaps. Both methods have been implemented in our system, though we only explained the gray-scale masks in Section 2.2. In some cases, we do find the transparent heat map visualization helps users get the image context more easily.

A.4 The Necessity of Layer-level Metrics

In this section, we present cases of important heads identified through the two *layer-level* head-importance metrics (Section 2.2.4.2) to validate their necessity. As we demonstrate in the examples, these metrics indicate the importance of a head within the *same layer*, as opposed to the *model-level* metrics that represent the head’s importance across *all layers*.

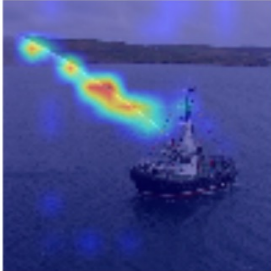
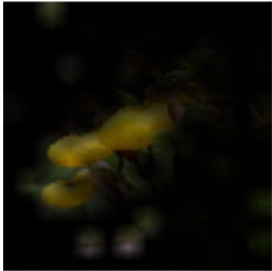
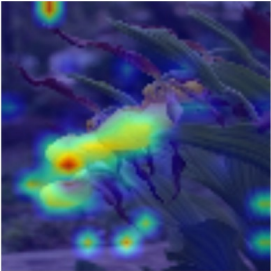
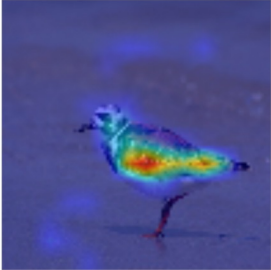
	<i>original</i>	<i>transparency mask</i>	<i>heatmap overlay</i>
<i>fire boat</i>			
<i>yellow lady's slipper</i>			
<i>macaw</i>			
<i>ruddy turnstone</i>			

Figure A.10: Example visualizations of transparency mask and heatmap overlay of the attention weights.

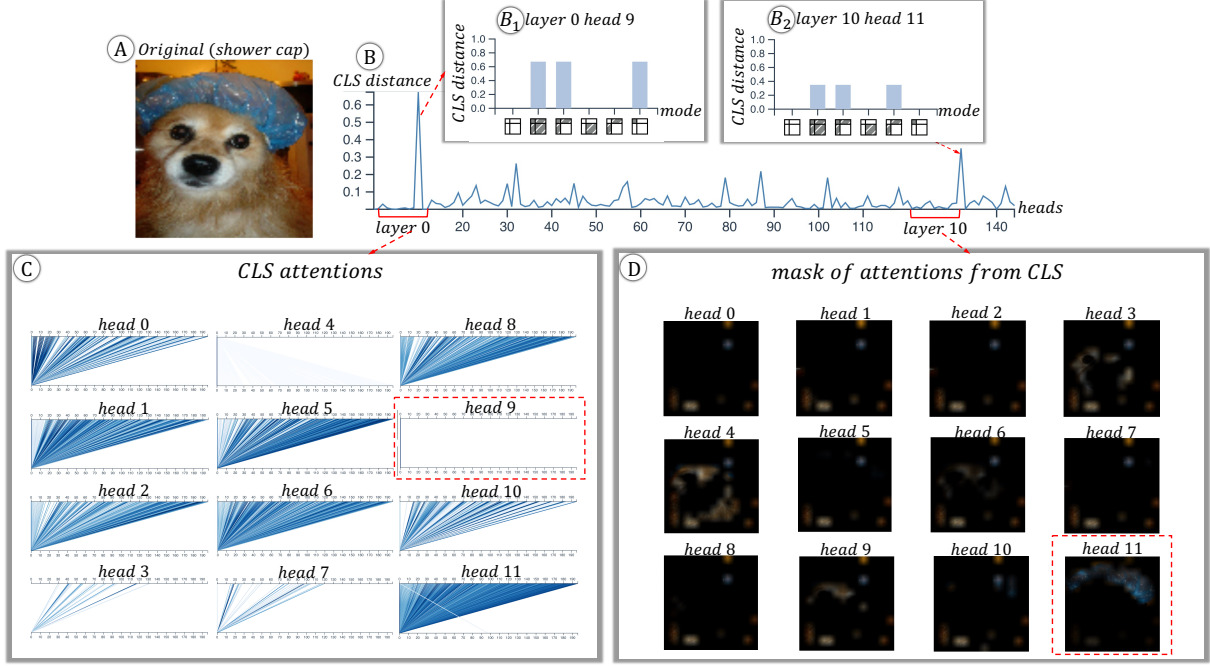


Figure A.11: Examples of important heads identified through the ‘CLS Distance’. Layer 0 head 9 is important because of the uniquely strong CLS $\rightarrow$ CLS attention. Layer 10 head 11 is important because of the unique semantics of `shower cap` captured by the CLS $\rightarrow$ *patches* attentions.

CLS Distance In Figure A.11, we use an example image labeled as `shower cap` to present the importance of heads with high CLS distances. From Figure A.11B, two heads are identified with the highest distance values. The CLS distances across different pruning modes are presented in Figure A.11B1 and B2 separately for each head. For layer 0, head 9 (Figure A.11B1), the CLS distance only rises when the CLS $\rightarrow$ CLS attention is pruned, whereas for layer 10, head 11 (Figure A.11B2), the distance only increases when the attentions between CLS and patches are pruned.

The importance of the two heads can be explained by scanning through all heads within the same layers. For layer 0 head 9, its importance comes from the CLS $\rightarrow$ CLS attention, as we learned from the partial pruning in Figure A.11-B1. From the CLS-related attentions of all 12 heads from layer 0 (Figure A.11C), head 9 is the only one with strong CLS $\rightarrow$ CLS attention. For layer 10 head 11, its importance comes from the attention between CLS and patches. Visualizing the extracted semantics by this part of the 12 head from layer 10 (Figure A.11D), we can see head 11 is the only one that captures the semantics of the `shower cap` through its CLS token.

Patch Distance The patch distance metric evaluates the layer-level uniqueness of each head regarding its attentions among patches. In Figure A.12A, two heads are identified with high

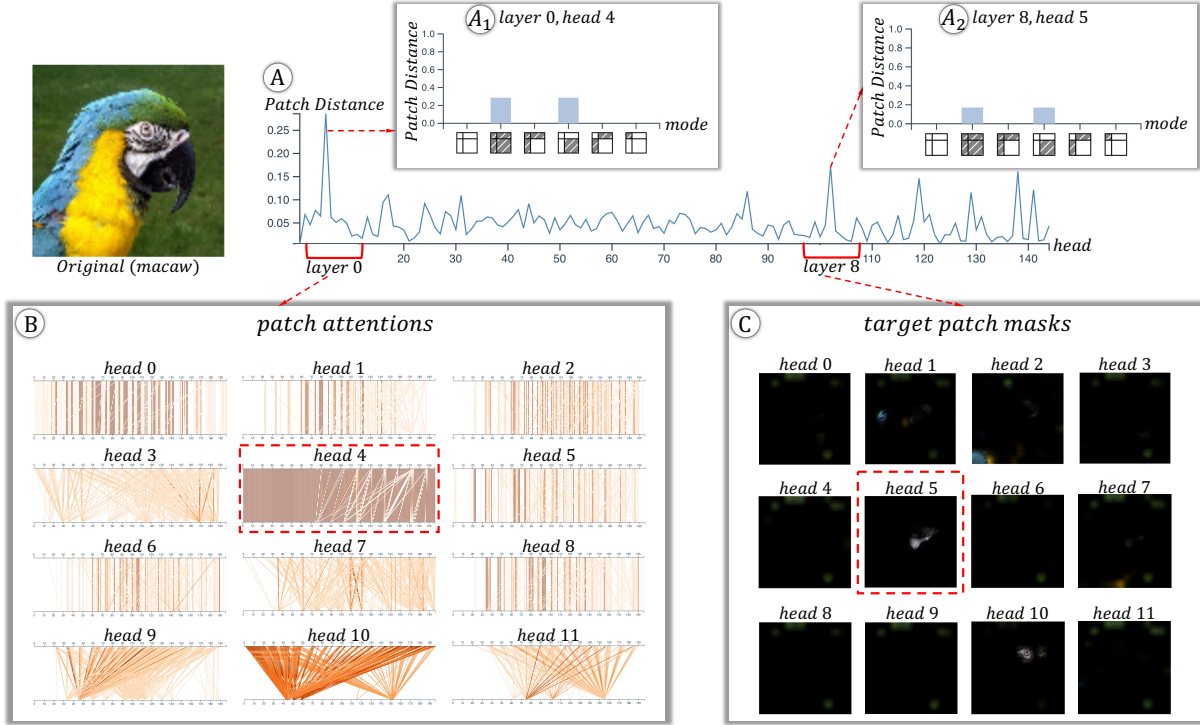


Figure A.12: Examples of important heads identified through the ‘Patch Distance’. Layer 0 head 4 is important because of its unique attention pattern that every patch attends strongly to itself. Layer 8 head 5 is important by the unique semantics of the *macaw*’s white face captured by the patch attentions.

patch distances, and both have high distances because the attentions among patches are pruned, which is indicated in Figure A.12A1 and A2. On the one hand, head 4 in layer 0 is important because of the uniquely high attentions each patch assigns to itself, as shown in Figure A.12B and discussed in Section 2.2.5.3. On the other hand, head 5 in layer 8 is important because of the unique image semantics captured by the patches in this head. This is explained in Figure A.12C, where the target image mask is visualized for each head, and each mask summarizes the overall distribution of attentions from the source patches. It is noticed that head 5 is the only head where patches focus mainly on the *macaw*’s face region.

A.5 Mean Attention Distance vs. Attention Strength

This work introduces *attention strength vector* to profile the spatial distribution of a head’s attention strength (Section 2.2.4.4). Previously, *mean attention distance* [76, 98] is the common

Algorithm 2: Mean Attention Distance

Data: Patch attentions $A_{patch} \in \mathbb{R}^{p^2 \times p^2}$, patch size pz

Result: Mean attention distance d for one head

```
1  $D \leftarrow \text{zeros}(p^2, p^2)$  ; // matrix of pixel distances between patches
2 def ComputeDistanceMatrix():
3     for  $i = 0$ ;  $i < p^2$ ;  $i++$  do
4          $x_0 \leftarrow \lfloor i/p \rfloor$ ;
5          $y_0 \leftarrow i \bmod p$ ;
6         for  $j = 0$ ;  $j < p^2$ ;  $j++$  do
7              $x_1 \leftarrow \lfloor j/p \rfloor$ ;
8              $y_1 \leftarrow j \bmod p$ ;
9              $d_{patch} \leftarrow \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$ ;
10             $D[i][j] \leftarrow d_{patch} \times pz$ ;
11 end def ComputeDistanceMatrix();
12  $D_{weighted} \leftarrow D \odot A_{patch}$  ; // element-wise multiplication
13  $\mathbf{d}_{weighted} \leftarrow (D_{weighted} \cdot \mathbf{1}_{p^2})^T$  ; // summation of weighted distances by columns
14  $d \leftarrow \text{mean}(\mathbf{d}_{weighted})$  ; // average of all the weighted sums
```

metric used to measure this. This section compares our *attention strength vector* with the *mean attention distance*.

A.5.1 Algorithm Description

For the computation of mean attention distance (introduced in earlier works), we use the implementation from [98]¹, and also provide the pseudocode in Algorithm 2. Essentially, the algorithm first computes a matrix of pixel distances between each pair of patches. Then, each distance value is weighted by the attention between each pair of patches. These weighted distances are then summed by each row, where the mean attention distance is the average over the sums of all rows.

The mean attention distance is used to evaluate the *span* of each head’s attention on images. Higher mean attention distances indicate that there are patches that assign high attention

¹Reference [link](#)

Algorithm 3: Attention Strength Vector

Data: Patch attentions $A_{patch} \in \mathbb{R}^{p^2 \times p^2}$

Result: Attention strength vector $\mathbf{s} = (s_0, s_1, \dots, s_{p-1})$

```
1  $\mathbf{s} \leftarrow \text{zeros}(p)$ ;
2  $\text{count} \leftarrow \text{zeros}(p)$  ; // counts of patches with corresponding hop neighbors
   /* iterate through each k-hop neighbor */
3 for  $k = 0$ ;  $k < p$ ;  $k++$  do
   /* iterate through each patch position */
4   for  $(i, j) \in \{0, 1, 2, \dots, p-1\}^2$  do
5      $P_{i,j} \leftarrow \text{PatchAttentions}(i, j, A_P)$  ; // attentions from  $(i, j)$  to all patches
6     if  $k == 0$  then
7        $\mathbf{s}[k] += P_{i,j}[i, j]$ ;
8        $\text{count}[k] += 1$ ;
9     else
10       $(x_0^k, x_1^k, y_0^k, y_1^k) \leftarrow \text{BoundingBox}(i, j, k)$   $(x_0^{k-1}, x_1^{k-1}, y_0^{k-1}, y_1^{k-1}) \leftarrow$ 
         $\text{BoundingBox}(i, j, k-1)$   $n_k \leftarrow (x_1^k - x_0^k + 1) \times (y_1^k - y_0^k + 1)$ ;
11       $n_{k-1} \leftarrow (x_1^{k-1} - x_0^{k-1} + 1) \times (y_1^{k-1} - y_0^{k-1} + 1)$ ;
12       $n_{\text{neighbors}} \leftarrow n_k - n_{k-1}$  if  $n_{\text{neighbors}} > 0$  then
13         $s_k^{(i,j)} = \frac{[\text{sum}(P[x_0^k:x_1^k, y_0^k:y_1^k]) - \text{sum}(P[x_0^{k-1}:x_1^{k-1}, y_0^{k-1}:y_1^{k-1}])]}{n_{\text{neighbors}}}$ ;
14         $\mathbf{s}[k] += s_k^{(i,j)}$ ;
15         $\text{count}[k] += 1$ ;
16      else
17        continue
   /* average strength over all positions */
18   if  $\text{count}[k] > 0$  then
19      $\mathbf{s}[k] /= \text{count}[k]$ ;
20   else
21      $\mathbf{s}[k] = 0$ ;
```

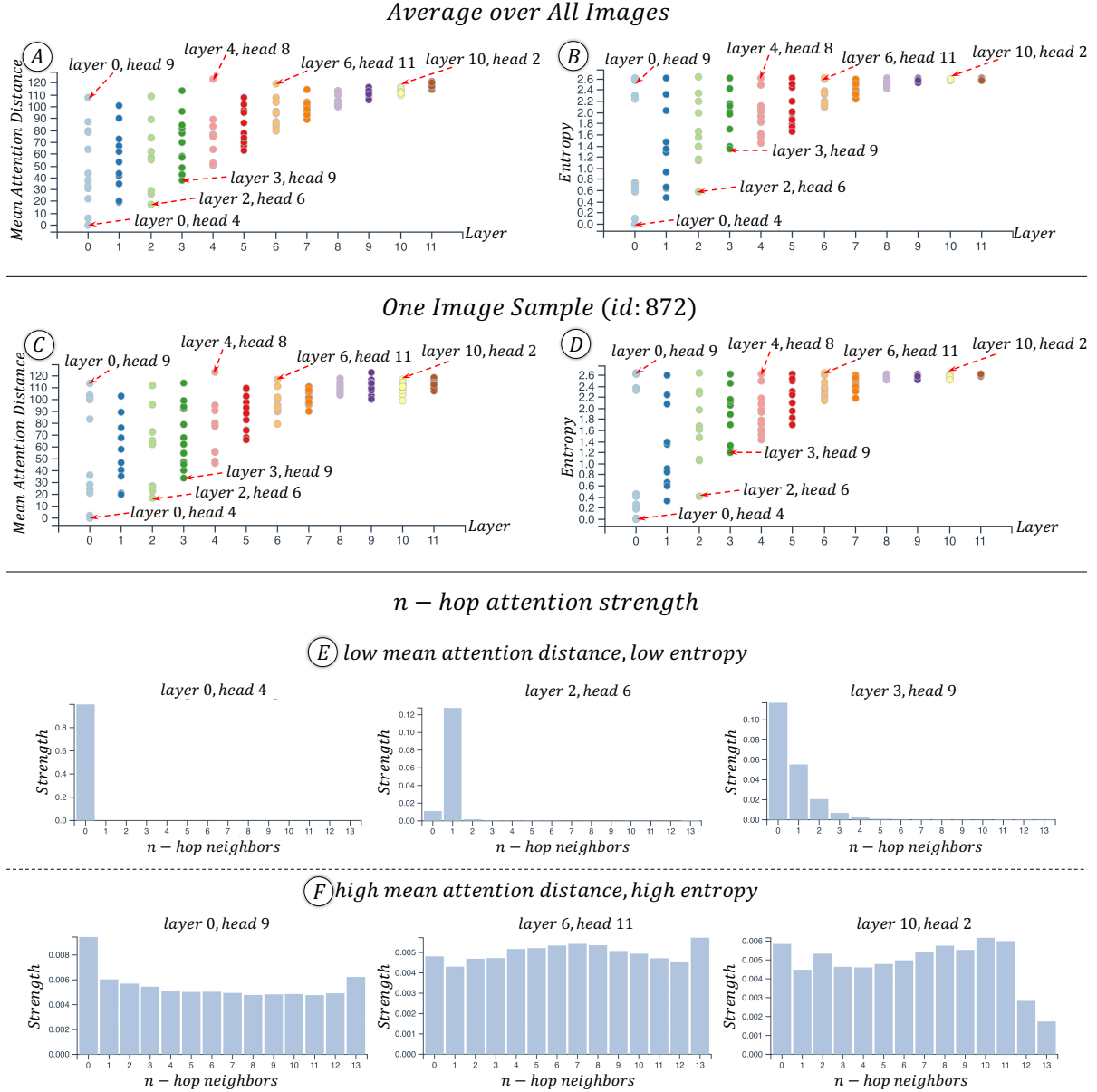


Figure A.13: Overall comparisons between the n-hop attention strengths proposed in our work and the mean attention distance method. We plot the mean attention distances and the attention strength entropies against the layers, and found high consistency among heads with the highest and the lowest measures for both the average values over all images (A, B) and values of a random image example (C, D). The heads with low values make sense for both metrics, as the highest attention strengths are concentrated at low hops of neighbors (E). However, heads with high mean attention distances don't have the global attention strength distribution as expected (F), which contradicts the assumption that these heads attend more globally (in fact, these heads attend both locally and globally as reflected by the flat distributions in F).

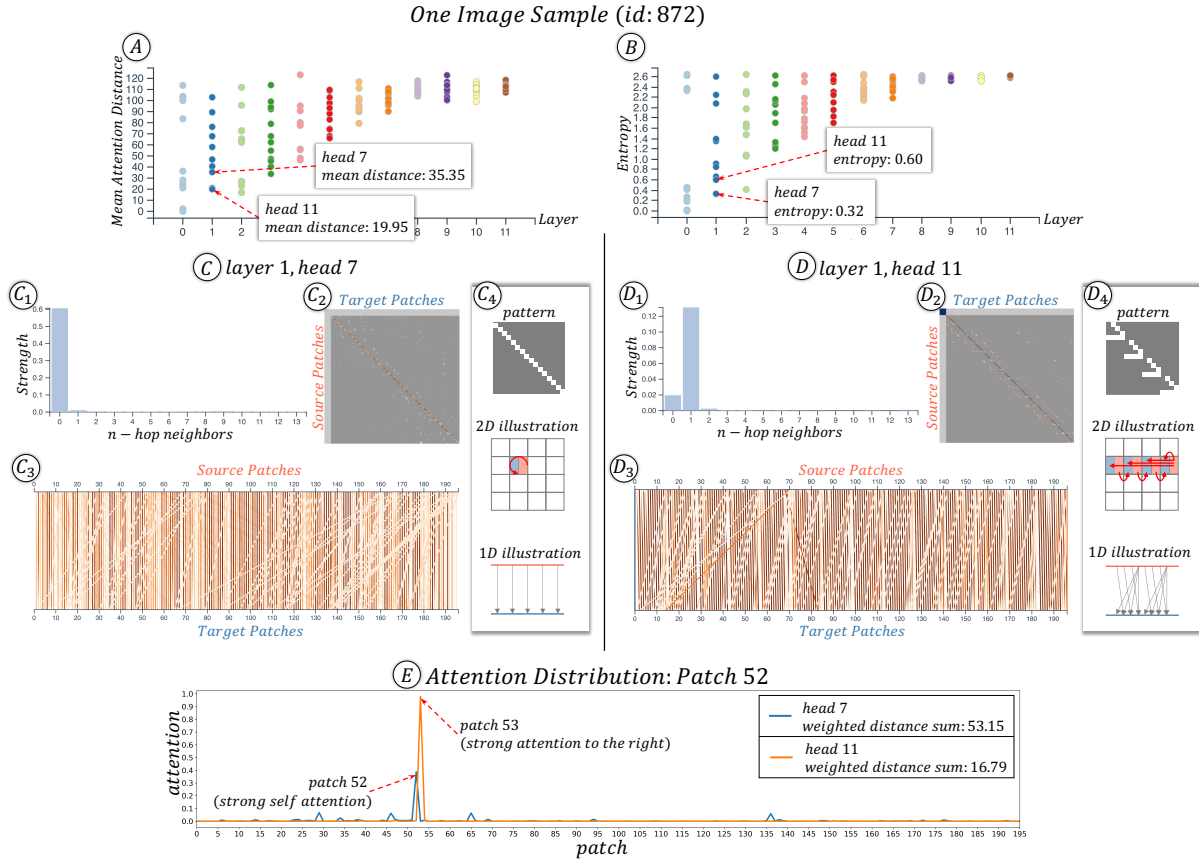


Figure A.14: An example of two heads that demonstrate the inaccuracy of mean attention distances. Head 7 and head 10 in layer 1 are ranked differently by mean attention distances and by attention strength entropies as shown in A and B. The attention strength distribution of head 7 (C1) indicates that it focuses mostly on 0-hop neighbors, which also means the strong self-attention pattern summarized in C2-C4. Head 11 focuses mainly on 1-hop neighbors (D1), and patches attend strongly to the right patch in the attention patterns (D2-D4). However, head 7 has a higher mean attention distance than head 11 even though head 11 overall attends farther away. This contradiction is explained through the distribution of attentions from an example patch in E. Despite that patch 52 has the strongest attention to itself for head 7, there are other much smaller attentions with large distances away from patch 52 (the small peaks in the blue curve), and will contribute more to the weighted distance sum.

Algorithm 4: Function Definitions

```
1 def PatchAttentions( $i, j, A_{patch}$ ):  
    /* extract attentions for patch ( $i, j$ ) */  
2      $row \leftarrow i \times p + j$ ;  
3      $P \leftarrow A_{patch}[row].reshape((p, p))$ ;  
4     return  $P$   
  
5 def BoundingBox( $i, j, k$ ):  
    /* return the bounding box centered at patch ( $i, j$ ), extending to  $k$  patches away */  
6      $x_0 \leftarrow \max(0, i - k)$ ;  
7      $x_1 \leftarrow \min(p - 1, i + k)$ ;  
8      $y_0 \leftarrow \max(0, j - k)$ ;  
9      $y_1 \leftarrow \min(p - 1, j + k)$ ;  
10    return ( $x_0, y_0, x_1, y_1$ )
```

values to patches farther away. A uniform pattern discovered by the previous works [76, 98] is that heads in lower layers attend both locally and globally, whereas heads in higher layers only attend globally. However, this metric overly aggregates the spatial information and falls short of presenting lower-level spatial details (See details in Section A.5.2).

The computation of attention strength vectors is elaborated in Section 2.2.4.4, and here we provide the pseudocode for its implementation details in Algorithm 3.

A.5.2 Method Comparison

We compare the overall distribution of the entropies of attention strength vectors, and the distribution of mean attention distances for all heads across different layers (Figure A.13).

First, high consistency is discovered among heads with the highest and lowest values in each layer. Specifically, heads with the lowest mean attention distances in each layer are also heads with the lowest entropies, and heads with the highest mean attention distances in each layer are also heads with highest entropies. In Figure A.13A and B, each head’s mean attention distance and entropy are averaged over all sampled images. The four heads at the top of layers 0, 4, 6, and 10 in A are also at the top of the corresponding layers in B, whereas the three example heads at the bottom of layers 0, 2, and 3 in A are also at the bottom in B. The same situation

happens with Figure A.13C and D, where the statistics are plotted for one image sample.

The attention strength vectors provide detailed explanations of the consistency between the two methods. Heads with low values in both mean attention distance and entropy can be explained by the extreme left-skewness of the attention strength vectors, as these heads mostly attend to closer neighbors. This is presented by three example heads in Figure A.13E. Heads with both high values, however, are more unintuitive. As shown in Figure A.13F, although these heads attend more to neighbors farther away, they *still* assign much attention to closer neighbors. The more balanced distribution of attention strengths explains the high entropy values but also means the head is attending both locally and globally, which challenges the accuracy of the high mean attention distances.

We further explain the inaccuracy of mean attention distances through an example of two heads ranked differently by the two metrics. In Figure A.14A and B, heads 7 and 11 of layer 1 are identified to have disagreement between mean attention distances and attention strength entropies. Head 7 has a higher mean attention distance, whereas head 11 has a higher entropy. From Figure A.14C, we learn that head 7 mostly attends to its zero-hop neighbors, corresponding to the attention pattern that each patch attends strongly to itself. Meanwhile, head 11 attends mostly to its one-hop neighbors, as its attention pattern is each patch attends strongly to its right neighbor. Head 11 has a higher entropy because it has a more balanced attention strength vector than head 7. However, it is counter-intuitive that head 7, with mostly zero-hop attentions, could have a higher mean attention distance than head 11, with mostly one-hop attentions.

Such a contradiction is caused by the defects in the computation of mean attention distances. Specifically, the pixel distance between patches is too dominant in its multiplication with the attention weight. When two patches are sufficiently far apart, the multiplication will be large even when the attention is minute. It can be observed that head 7 contains many small attentions that target random patches far away from the source locations (in Figure A.14C3), whereas this seldom happens for head 11 (in Figure A.14D3). In Figure A.14E, we elaborate this dominant effect by visualizing the attentions from an example patch 52 to all patches for both head 7 and head 11. For head 11, patch 52 assigns its largest attention weight to patch 53, while the attentions to other patches are almost zero. Head 7 assigns the largest weight to itself and small attentions to several locations far away. These small attentions are negligible compared to the strong self-attention but are powerful enough when multiplied by the large distances and

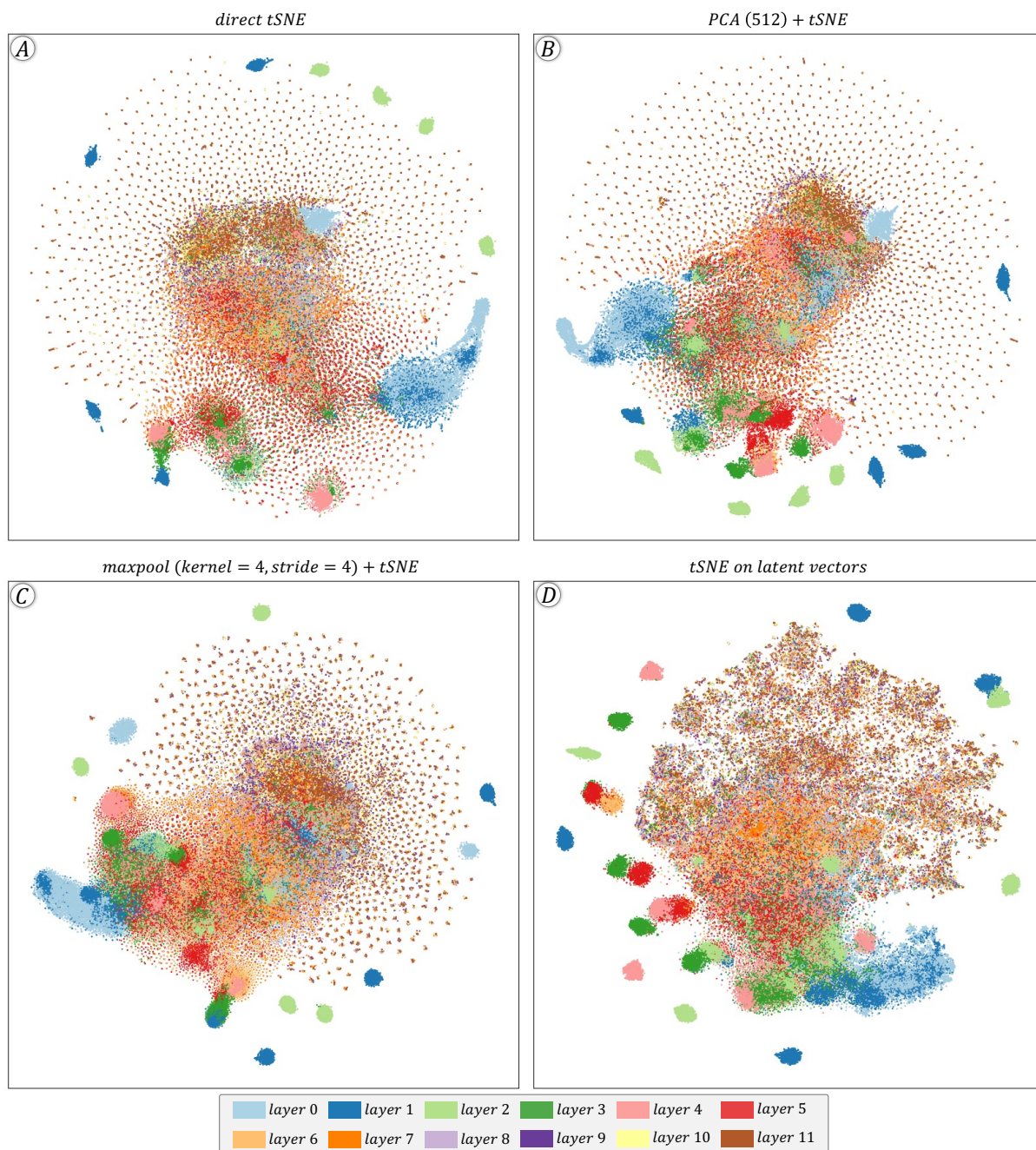


Figure A.15: Dimensionality reduction results with four different methods: direct t-SNE on raw attentions (A), t-SNE after PCA to 512 dimensions (B), t-SNE after max pooling the attention maps (C), and t-SNE of the latent vectors extracted from a well-trained autoencoder (D).

combine to a weighted distance sum (53.15) much larger than head 11 (16.79).

In conclusion, the mean attention distance is inaccurate because of the dominance of patch distances. The n -hop attention strength method avoids this problem by separating the attention weights from the distance information, where different neighborhood categories indicate the distance, and the average attention weights are computed for each neighborhood.

A.6 Comparing Dimensionality Reduction Methods

This section compares four different dimensionality reduction methods of the patch attention maps we have tried while coming up with the autoencoder-based solution (detailed in Section 2.2.4.5). We use the attention heads of the ViT-B_16-224 model to showcase the results, where each attention map is a 196×196 matrix.

First, we use t-SNE directly on these flattened raw attentions. For the second method, we first use PCA to project the raw attentions to 512 dimensions to be comparable with the latent dimensions and then use t-SNE on the 512D vectors. Similarly, the third method also preprocesses the raw attention to smaller dimensions before t-SNE but uses max pooling on the 196×196 map instead of PCA, where we set the kernel and the stride to be 4. Finally, the fourth method is our proposed solution of applying t-SNE projection on the 512D latent vectors extracted from the trained autoencoder.

Figure A.15 presents the results from the four methods, where each point represents the attention map of one head and is colored by the layer of the head. In general, all four results show the trend of “content-agnostic heads in lower layers, content-relevant heads in higher layers”, as the clusters in lower layers have more apparent separations, whereas clusters in higher layers are more mixed up.

The dimensionality reductions are expected to effectively gather heads with similar attention patterns and separate different ones. By comparing the four methods’ results, we find that the autoencoder-based method achieves the most effective dimensionality reduction. More isolated clusters are identified in Figure A.15D, containing the same heads for all images. As presented in Section 2.2.5.3, each cluster represents a unique attention pattern of a specific head. In contrast, many of these unique patterns cannot be identified in the other three results (Figure A.15A,

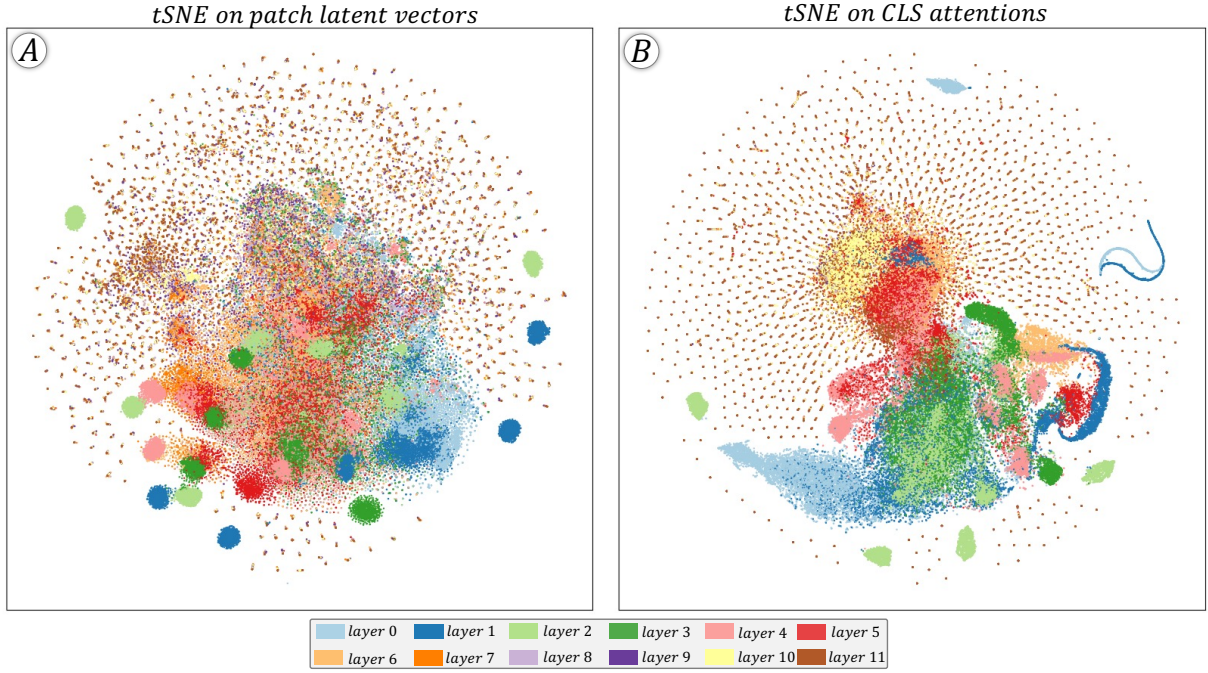


Figure A.16: Dimensionality reduction results of the patch attention latent vectors (A) and the CLS attentions (B) from the images sampled from CIFAR10.

B, and C), as heads with different patterns are mixed up together. This demonstrates that the latent vectors learned by the autoencoder capture more essential attention patterns and make the dimensionality reduction easier. The other three methods fail to identify the same number of patterns due to the high dimensions of the attention maps.

A.7 Explorations with the CIFAR10 Dataset

To validate our findings about ViTs across different datasets, we further explored a ViT model trained on the CIFAR10 dataset [206] with our visual analytics system. To achieve this, we first downloaded the ViT-B_16-224 model pre-trained on ImageNet, and fine-tuned the model using the training images of CIFAR10. Similar to the exploration with ImageNet, we randomly sampled 1000 test images of CIFAR10 and plugged them into our system.

Most of the exploration findings from CIFAR10 are very similar to what is shown in Section 2.2 using the ImageNet dataset. Here, we briefly demonstrate the head attention patterns from this dataset to show how its attention pattern trend is similar to that of ImageNet across different

attention layers of the ViT.

Figure A.16 presents the dimensionality reduction results of the patch (Figure A.16A) and CLS (Figure A.16B) attentions. Specifically, for patch attentions, we use the same autoencoder-based method to learn a latent vector for each patch attention matrix, and perform t-SNE on the latent vectors of all images. For CLS attentions, we directly use t-SNE on the attention weights. From Figure A.16, we found the following attention pattern trend across layers. *First*, the isolated clusters of heads are from lower-layer, i.e., the clusters of blue and green points. The higher layer heads (in yellow and brown) are mostly scattered everywhere. *Second*, each of the isolated clusters in both Figure A.16A and Figure A.16B is composed of the same lower-layer head from different images, which indicates that these lower-layer heads have fixed functionalities that are agnostic to the image contents. The two observations are consistent with what we have seen from the ImageNet dataset, i.e., the lower layers contain more content-agnostic heads (forming into isolated clusters) and the higher layers contain more content-relevant heads (with no obvious clusters).

Appendix B

Appendix for Chapter 4

B.1 More Results on Image Data Exploration

This section delves into additional examples showcasing the utilization of our *Co-Occurrence Graph* in the exploration of ImageNet [173] and COCO [174] datasets. These cases serve to underscore the versatility of the system in effectively navigating diverse image classes.

B.1.1 Exploring Different Species in ImageNet

While exploring the `cancer_irroratus` (Atlantic rock crab) images, we observed that our *Co-Occurrence Graph* adeptly encapsulates the natural habitat of this crab species. Notably, in the overview depicted in Figure B.1a, the word `rock` immediately emerges as the prominent one, with the highest co-occurrence with the word `crab`. Clicking on the node for `rock` grants us access to image segments that are strongly associated with this word. Several illustrative segments are presented in Figure B.1b, alongside the corresponding source images and their respective captions. From the captions, it’s noteworthy that the word `rock` is identified independently rather than as part of the phrase “rock crab”. This suggests that the captioning model recognizes crabs from the images, but it does not know that they are rock crabs (i.e., the model is not capable of differentiating different types of crabs). The segments that exhibit a strong response to the word `rock` correspond to the rocky terrains frequently inhabited by `cancer_irroratus`, reflecting the typical habitat of this crab species.

Next, by scrutinizing `green_snake` images, our system unveils the prevalent living habits of this species. In the *Co-Occurrence Graph* view for `green_snake` (Figure B.2a), we initially observed that the word `green` most frequently co-occurs with `snake`, indicating the snakes’ color. Following `green`, the three words that emerge with notably high co-occurring frequencies are

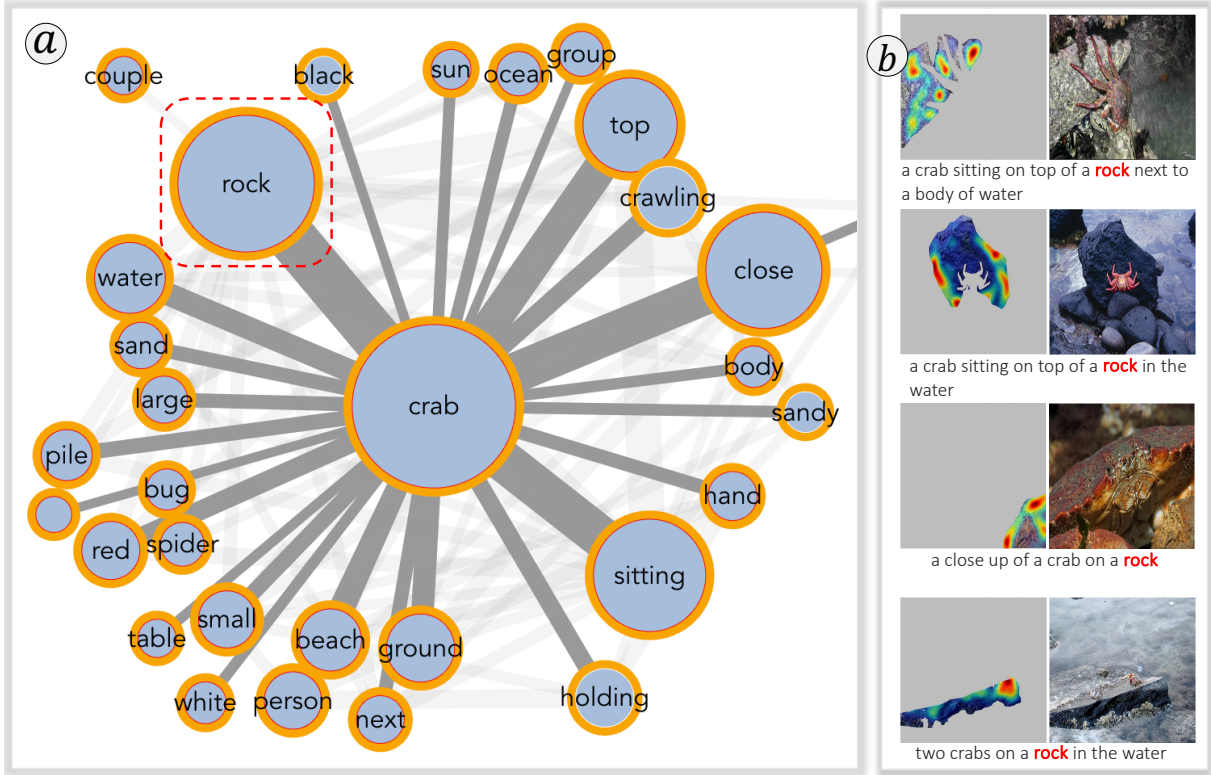


Figure B.1: (a) For the captions of `cancer_irroratus` images, `rock` and `crab` are the dominant and most co-occurring features, reflecting the fact that this type of crab prefers a rocky habitat. (b) Some example segments responding strongly to the word `rock`.

`branch`, `curled`, and `tree`. This suggests that `green_snakes` are commonly found in proximity to a `tree_branch` and often exhibit a `curled` posture. A more in-depth examination of the image segments that strongly respond to the words `branch` and `curled` (Figure B.2b and Figure B.2c) substantiates the prevalence of this distinctive posture among `green_snakes`.

We have also investigated the images of two similar species of birds, `black_stork` and `ptarmigan`, and compared their subtle differences using our system. From the exploration, we have several observations. **First**, as we have repeatedly noticed (in the `tench` and `crab` cases), the pre-trained BLIP is not able to differentiate different subtypes of birds and use the word `bird` to describe both species (Figure B.3). **Second**, `black_storks` have long beaks and inhabit mostly around river or marshy areas, whereas `ptarmigans` have much shorter beaks and inhabit mostly on the highlands. These differences can be easily observed through the respective *Co-Occurrence Graph*. As shown in Figure B.3, the most frequently co-occurred words for `black_stork` images are `water` and `beak`, while the most frequently co-occurred words for `ptarmigan` are `rock` and `snow`. **Third**, the words `flying`, `blue` and `sky` also appear frequently

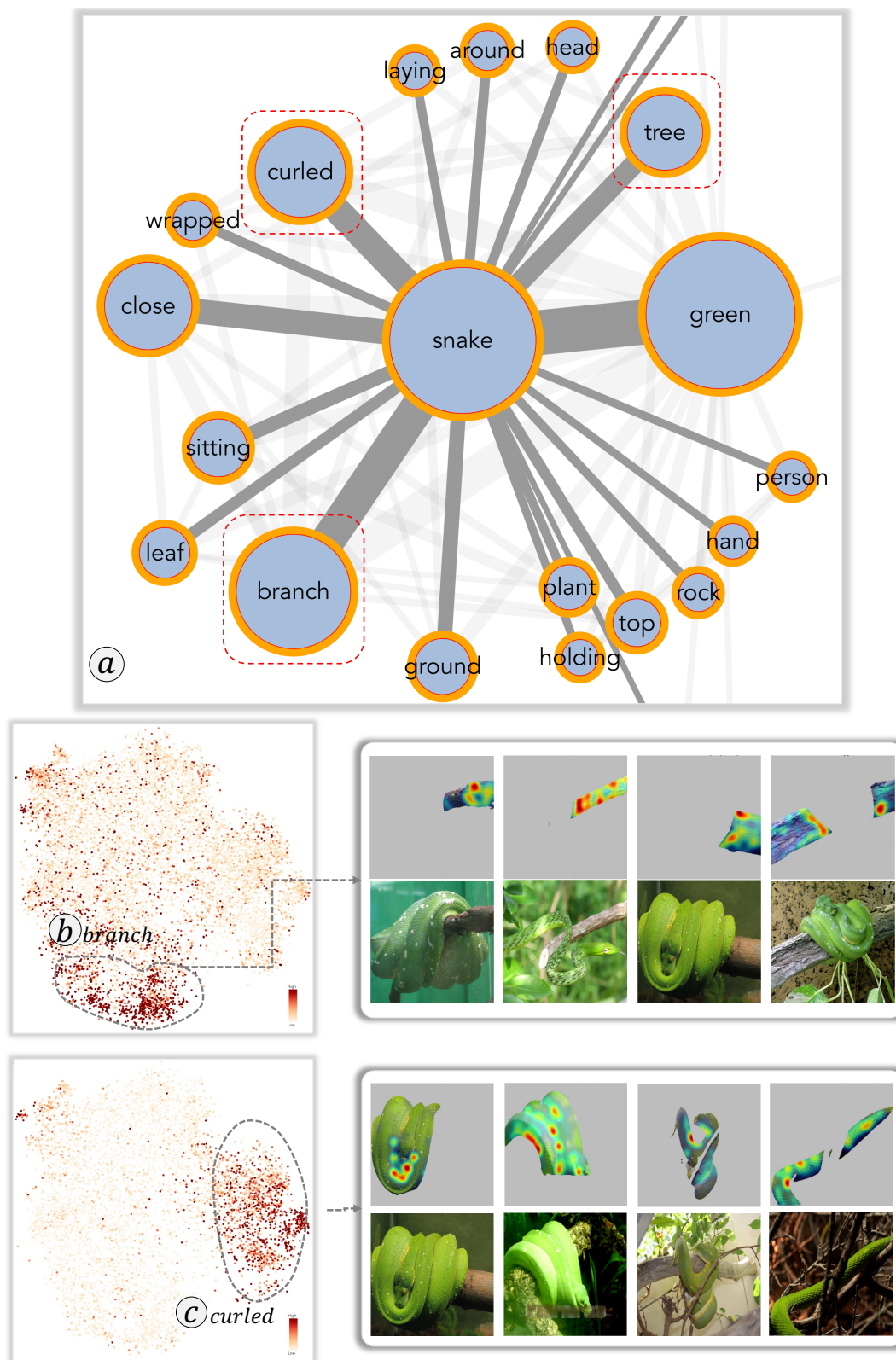


Figure B.2: (a) The dominant words in the captions for **green_snake** images are **green**, **branch**, **tree**, and **curled**. These words and their co-occurrence relationships depict the living habits and main postures of this type of snake. (b) and (c) show some example segments that strongly respond to the word **branch** and **curled**, respectively.

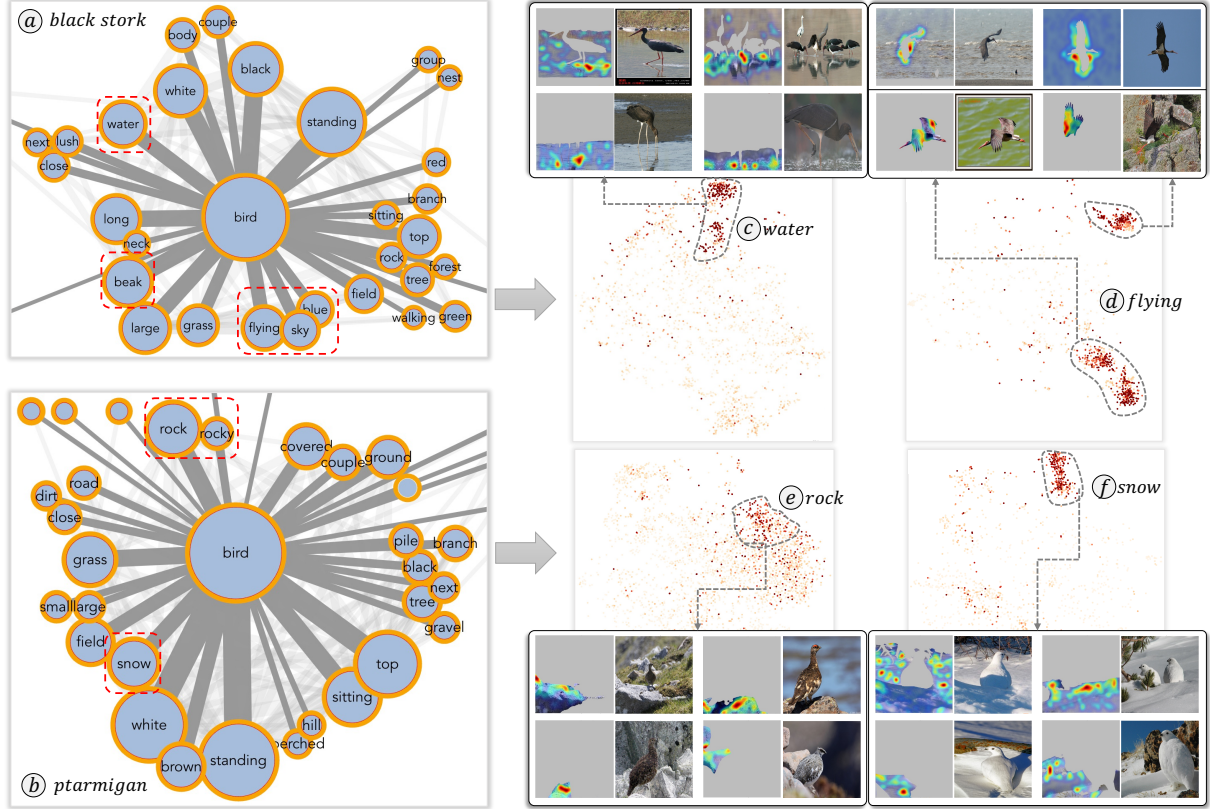


Figure B.3: Using our system to compare two similar species of birds: **black_stork** (a, c, d) and **ptarmigan** (b, e, f). Our visualizations could reveal the difference of the two birds in their living environment and behavior patterns.

for **black_stork**, but are not observed from the node-link diagram for **ptarmigan**. This indicates another difference between the two species: **black_storks** are more often found flying (Figure B.3d), while **ptarmigans** prefer staying on the ground. **Lastly**, by exploring the segments extracted from **ptarmigan** images that respond strongly to the words **rock** and **snow**, we also noticed that the plumage color of **ptarmigan** changes from brown to white in different seasons (i.e., seasonal camouflage) to better hide themselves into the environment. As shown in Figure B.3e and Figure B.3f, the **ptarmigan** with **rock** and **snow** background are in brown and white colors, respectively.

B.1.2 Potential Biases with Human-Related Objects in COCO

Next, we focus on several human-related objects from COCO images to analyze the co-occurring features within each object.

We first attempted to identify potential gender biases in images of various sport equipment, i.e., **skateboard**, **surfboard**, and **skis** (Figure B.4), and vehicles, i.e., **car** and **motorcycle**

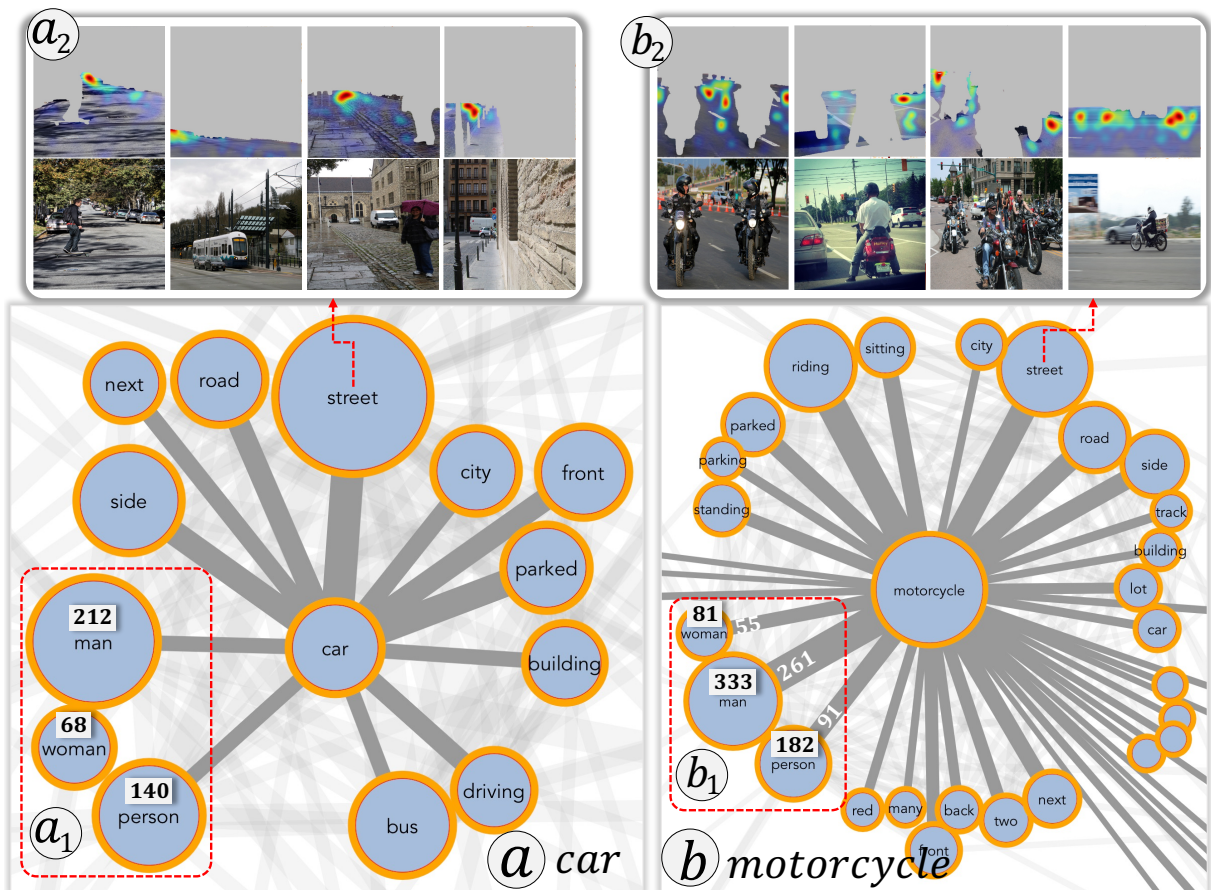
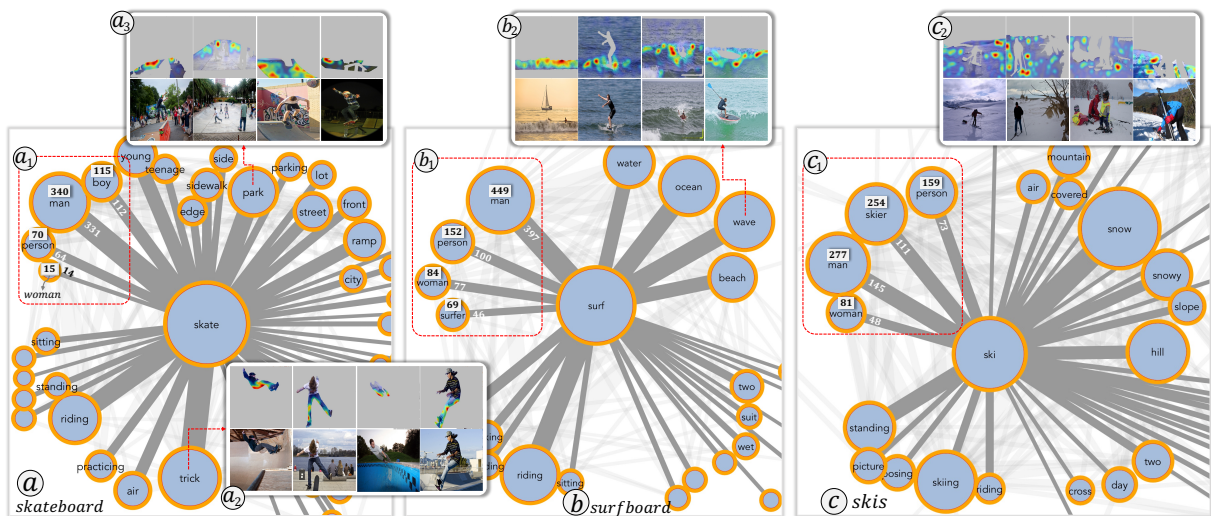


Figure B.5: Exploring the feature co-occurrences of several vehicle objects from COCO: (a) `car`, (b) `motorcycle`.

(Figure B.5). From the *Co-Occurrence Graph* view, we observed a common pattern among these objects where the frequencies of **woman** and **person** were significantly lower than that of **man** (see Figure B.4-a1, b1, c1, and Figure B.5-a1, b1). Notably, not only did **man** have the highest occurrence in these images (see the corresponding nodes’ size), but it also surpassed **woman** and **person** in terms of co-occurrence frequencies with the sport or vehicle objects in focus (see the corresponding links’ width). One extreme example is found within the **car** images, where there was no link between **car** and **woman** (Figure B.5-a1). The exact frequency values are annotated above each node and link, providing a basis for these comparisons. These imbalanced gender occurrences in the images highlight potential imperfections in image sampling, which may lead an ML model to exhibit biased behaviors in gender-related tasks.

Meanwhile, we have also explored other contextual features that prominently co-occurred with the target objects. For example, among the **skateboard** images, the most commonly co-occurring words include **trick** and **park**. These words are associated with the actions of skateboarders (e.g., performing a **trick**) and the skate-park environment (Figure B.4-a2, a3), shedding light on the dominant human activities and settings related to **skateboard**. Similar dominance of environmental features has also been identified in **surfboard** images (e.g., **wave** in Figure B.4-b2) and **skis** images (e.g., **snow** in Figure B.4-c2). For the vehicle images, like **car** and **motorcycle**, the most frequent co-occurring feature is **street** (Figure B.5-a2 and b2), which indicates that these vehicles are mostly depicted in urban street settings.

B.1.3 Exploring Different Scenes in Places365

We further extend our exploration to Places365 [161], an image dataset composed of different scenes, and analyze the features and their co-occurrences in each scene category.

While exploring the **hospital** images, we first noticed that **building** is the most dominant word among the image captions, which is shown as the largest node in the *Co-Occurrence Graph* view of Figure B.6. However, the word **hospital** is not present among the nodes. Examining the image segments attended by **building** (Figure B.6a), we discovered that most of the buildings in the images have a rather general appearance so it is difficult to specify their functionality. Next, we identified some features that highly co-occur with **building**, such as **car**, **parked**, and **front**. We investigated the segments mostly attended by **car**, and figured out that the cars are often parked in front of the hospital building. Such a strong co-occurrence may be utilized by a downstream model to differentiate hospital buildings against other scenes, and thus requires

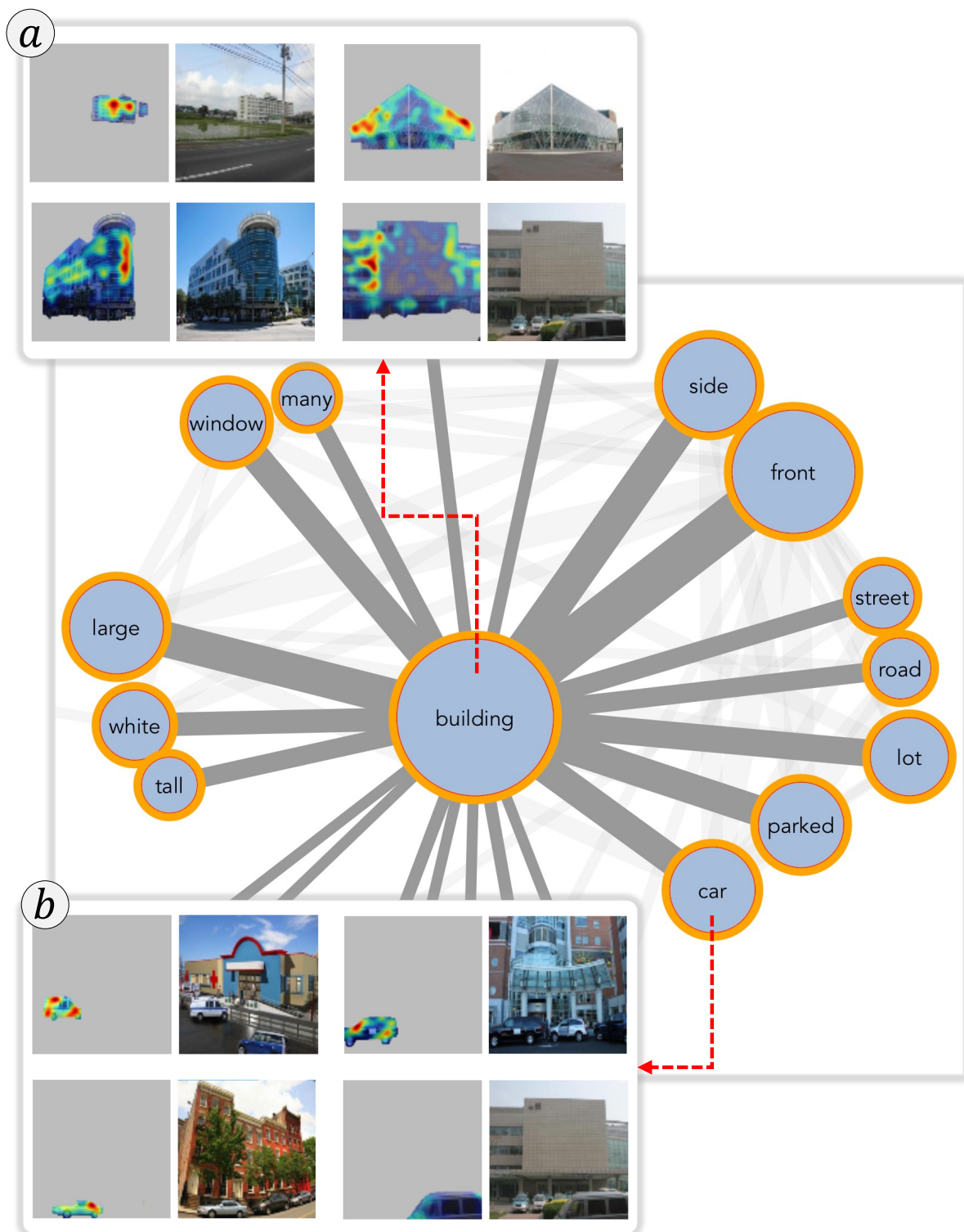


Figure B.6: Within **hospital** images, the **building** of the hospital is a dominant feature, and it co-occurs highly with the **cars** parked in the front.



Figure B.7: Comparison between the **kitchen** and the **living room** categories. Features such as **sink** and **stove** are found highly co-occurred with **kitchen**, while **couch** and **chair** co-occur a lot with **living room**.

more attention.

We also used our exploration tool to compare two scenes, **kitchen** and **living room**, and identified their most distinguishable features. In the *Co-Occurrence Graph* view of these two categories, we first found that the captions are able to distinguish the scenes correctly, as demonstrated by the dominant nodes **kitchen** in Figure B.7a and **living room** in Figure B.7b. Then, we focused on the co-occurred features with these major categories, and discovered several features that are exclusive to each category. For example, **sink** and **stove** are often found in kitchens and thus have high co-occurrences. The image segments corresponding to these two

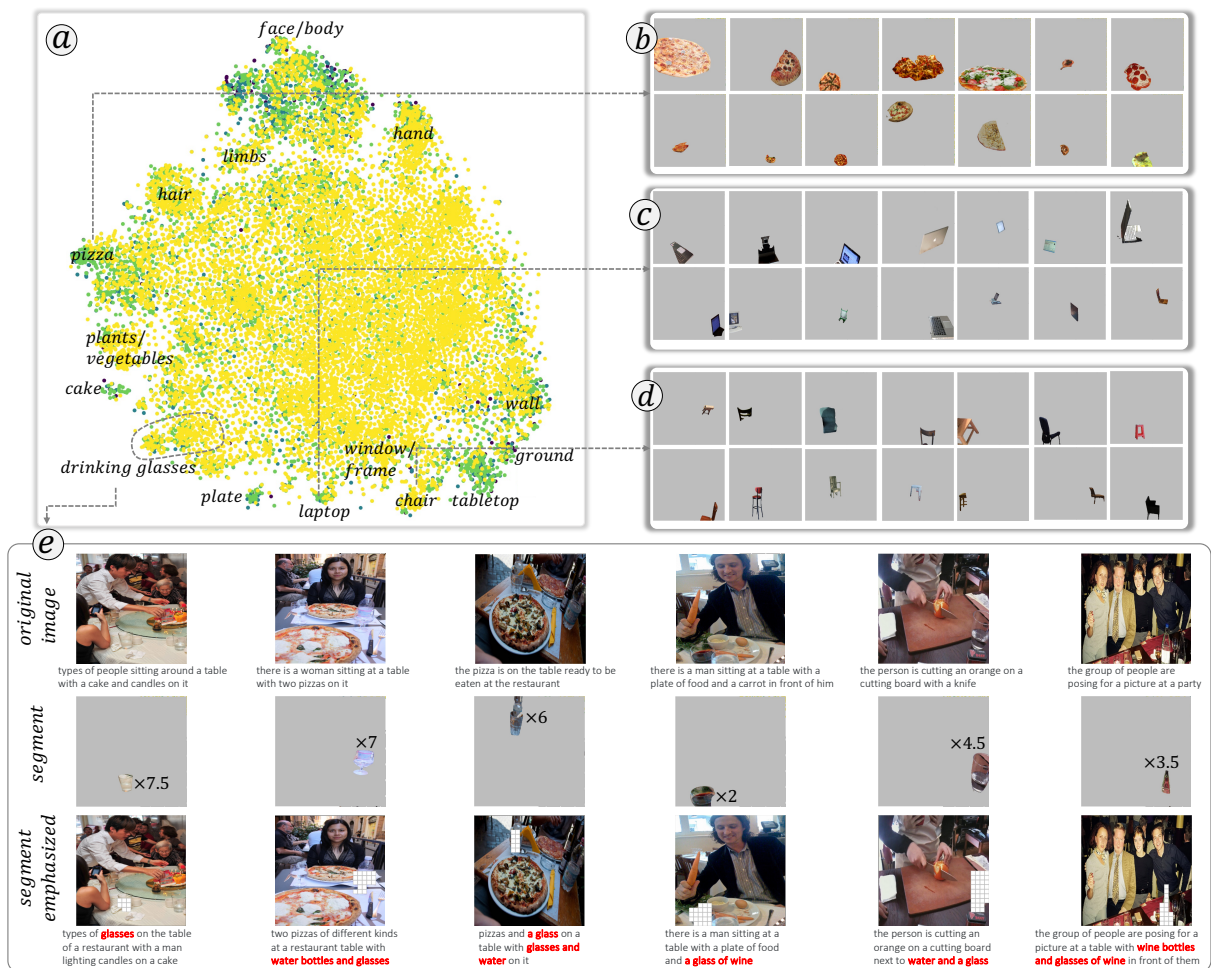


Figure B.8: (a) The clusters of image segments from the `dining-table` images. (b, c, d) Lasso-selection results of the `pizza`, `laptop`, and `chair` segments. (e) Steering the image captioning model to generate descriptions for `drinking-glasses` by emphasizing the corresponding segments.

words are shown in Figure B.7-a1 and a2. On the other hand, `couch` (Figure B.7-b1) and `chair` (Figure B.7-b2) are common features in living rooms, so they are the unique co-occurred features in the `living room` category. Such a comparison between different scene categories can highlight the features unique to each category, which might be used by ML models to classify the scenes.

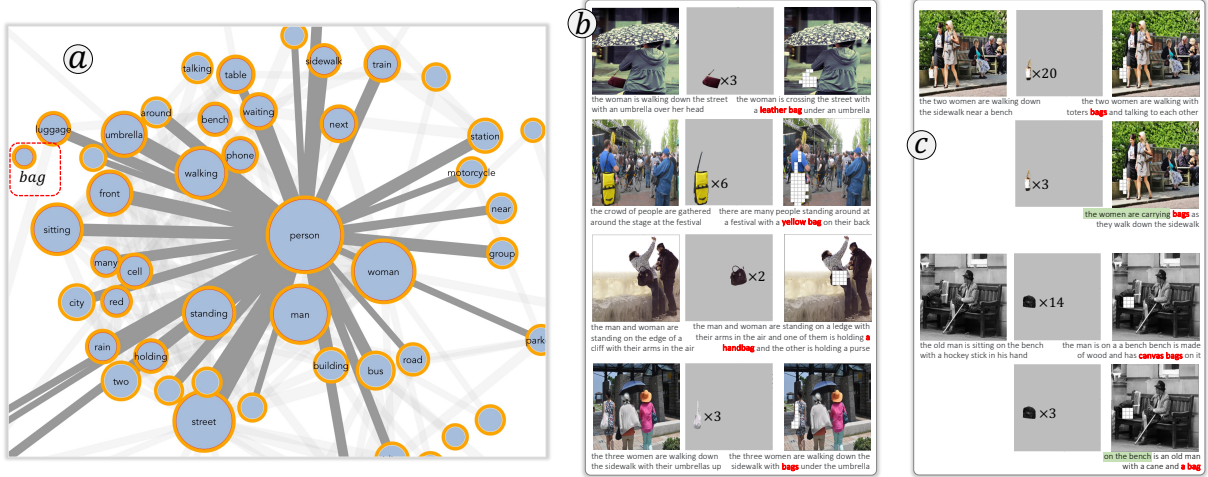


Figure B.9: (a) The *Co-Occurrence Graph* view of **handbag** images. (b) Steering the image captioning model by adjusting segments' weight only. (c) Steering the image captioning model by adjusting the segment's weight and modifying the starting prompts.

B.2 More Results of Caption Coverage and Steering

This section includes more explorations with our system on caption coverage and caption steering, in addition to the **tench** and **person+car** cases that are shown in Sections 4.1.5.2 and 4.1.5.3.

First, we examined the COCO images that feature both **person** and **dining.table**. Several clusters with common features can be identified through the *Image Segment* view (Figure B.8a). These clusters include human-related features, such as **face**, **body**, and **hair**. They also include features associated with dining tables, such as **pizza**, **laptop**, and **chair** (shown in Figure B.8b-d). Among these clusters, we noticed that many image segments within the **drinking.glasses** cluster are not sufficiently covered by the captions, but this feature is an essential object on a **dining.table**. Therefore, we target to steer the image captioning model to generate new captions describing this feature. Several caption steering results are shown in Figure B.8e. With the guidance of the **drinking.glasses** segments (the middle row), we are able to select and scale up the corresponding image patches and update the captions to generate the desired feature descriptions (the bottom row).

Next, we move on to explore another set of COCO images featuring both **handbag** and **person** objects. To our surprise, we discovered that our target feature, i.e., the **bag**, is rarely mentioned in the image captions and seldom co-occurs with other objects from the *Co-Occurrence Graph* view, as highlighted in Figure B.9a. As we delved deeper into these images and scrutinized the **handbag** segments, we realized that the **handbags** depicted in most of them are smaller

in scale compared to other objects, such as **person**, **bench**, and **umbrella**. As a result, these smaller **handbags** are often unnoticed during the caption generation process. In Figure B.9b, we present four example images and the corresponding segments, where we lay more emphasis on the **handbag** patches and regenerate the captions. Utilizing solely the segment scaling technique, we successfully ensured the inclusion of the word **bag** in all four images’ new captions. The two cases in Figure B.9c, however, require much larger scaling weights. With additional guidance from the starting prompts, we are able to generate desired captions with much smaller segment scaling weights.

B.3 Batch Processing Results of Caption Steering

In Section 4.1.5.3, we explained one case in which we applied a new prompt to a batch of images to generate new captions. Here, we provide more details to support that case. Specifically, we lasso selected a cluster of image segments representing the hat features in the **tench** images. After removing duplicated images and the images with **hat** in their captions, 214 images were identified. Then, we examine all 214 images to make sure there is at least one person in each image and they all contain hat-like segments. This can be completed efficiently by scanning through the small preview images in the system (just as Figure 4.1-b1). We changed the default prompt from “a picture of” to “the person is wearing” for these 214 images and regenerated their captions. Among the 214 new captions, 193 contain words describing the hat features (e.g., **hat/hood/beanie**), resulting in a success rate of 90.19%. We further went through all 193 successful images to make sure the **hat** features are indeed presented in each image.

Next, we inspected the 21 (214–193) failed cases. Among them, 12 images indeed have the hat feature but the updated prompt cannot help to generate descriptions for it in the captions. For each of the 12 images, we then applied the caption steering in two ways: (1) scaling up the hat patches with the original prompt “a picture of”, and (2) simultaneously scaling up the hat patches and using the new prompt “the person is wearing”. The experiment results are presented in Figure B.10. For the first method of scaling up the patches’ weight only, the model still failed on some images. However, it was able to generate descriptions for the hat features in all 12 images’ captions, when using the second method, i.e., adjusting both the prompt and

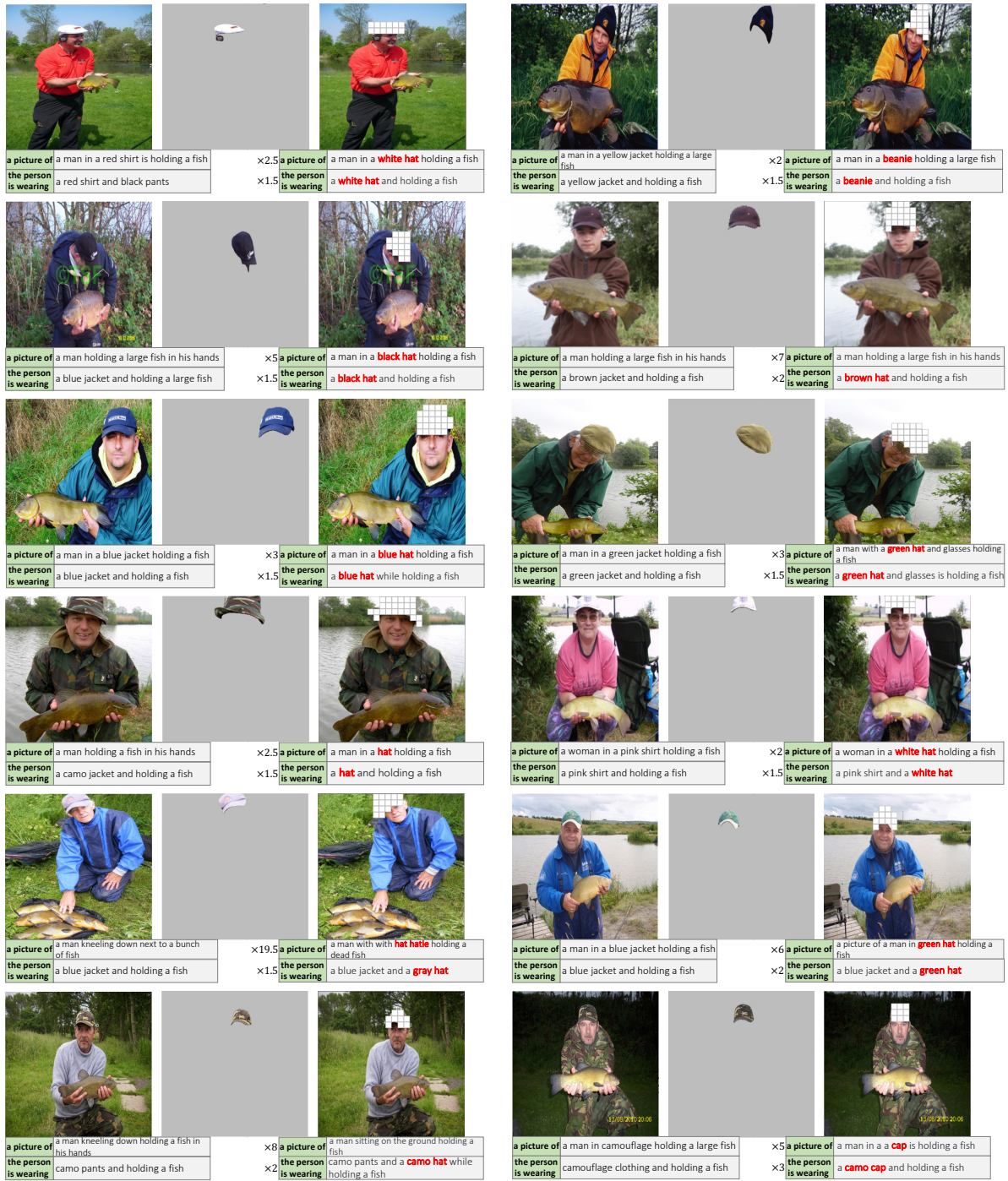


Figure B.10: The 12 images that failed to generate descriptions for **hat** features when applying prompt steering only. Explicitly enhancing the **hat** segments or using both prompt-engineering and segment enhancement successfully guide the model to generate descriptions for **hat** in the new captions.



Figure B.11: The extracted segments from these 9 images are very similar to hat features. However, they are actually not hat and we cannot steer the model to generate captions involving the word **hat**.

the patches’ weight. For the remaining 9 (21–12) images, we found that they actually do not contain hat features, but the SAM extracted features from them are very similar to a hat, as shown in Figure B.11. This detailed exploration explains why the model fails in some cases and further validates the good performance of our caption steering methods.

B.4 Grad-CAM Evaluation and Computation Details

In Section 4.1.4.2, we explained how Grad-CAM is computed (Figure 4.5) and used to disclose the association between image segments and caption words. This section provides more details to support the evaluation and computation of Grad-CAM.

B.4.1 Cross-Attention Evaluation

There are two groups of cross-attentions from the BLIP model (see Figure 4.2). One is from the LM part and the other is from the ITM part. Each group has 144 copies of the cross-attention matrix, since each part is a transformer with 12 layers and 12 heads. Aggregating the cross-attentions from the 12 heads of the same layer is commonly adopted in previous works [165] and

we employed the same operation. Therefore, we only need to compare the aggregated attentions from 12 layers. Additionally, we also want to show that using Grad-CAM is better than using cross-attention (CA) only. That is, the matrix C in Figure 4.5 is better than matrix A in reflecting the association between image features and caption words. These two perspectives generate four combinations: LM_CA, ITM_CA, LM_Grad-CAM, and ITM_Grad-CAM.

Figure B.12 shows the performance of the four methods. The horizontal axis denotes the 12 layers, the vertical axis presents the grounding accuracy (the higher the better). The grounding accuracy is computed by using the CA or Grad-CAM matrix on the RefCOCO+ dataset [205] and comparing the matrix-encoded association with the ground-truth association. Specifically, for each image-caption pair from the RefCOCO+ dataset, people have annotated which word is associated with what image features (visual objects). Considering this as the ground-truth, we can check if the CA or Grad-CAM matrix learned the correct attention between caption words and image patches. Eventually, we report the grounding accuracy, which is the ratio between the image-caption pairs that the captions accurately attend to the correct visual objects and all image-caption pairs. Based on the results shown in Figure B.12, the Grad-CAM computed from the ITM part of BLIP achieves the best performance at layer 7, and the accuracy value is much higher than other methods or layers. Therefore, we chose to use the Grad-CAM extracted from layer 7 of ITM in our visual analysis by default. However, as we have explained in Chapter 4, users can also flexibly switch to other methods/layers, using the widgets from the headers of the *Image Segment* view.

Focusing on the Grad-CAM from layer 7 of ITM, we then drilled down to individual heads of this layer to see their grounding accuracy. As shown in Figure B.13. Among the 12 heads of layer 7, head 1, head 6, and head 11 achieve better grounding performance than the other ones. The grounding accuracy of Grad-CAM averaged over all heads of this layer is also plotted for reference, which is on par with the best performance of individual heads. This result further supports us in aggregating the 12 heads from individual layers.

B.4.2 The Computation Details of Grad-CAM

To help people reproduce our experiments, we provide further details on the computation of Grad-CAM to support our explanations in Section 4.1.4.2. The details are illustrated in Figure B.14 and the computation process can be elaborated with the following steps.

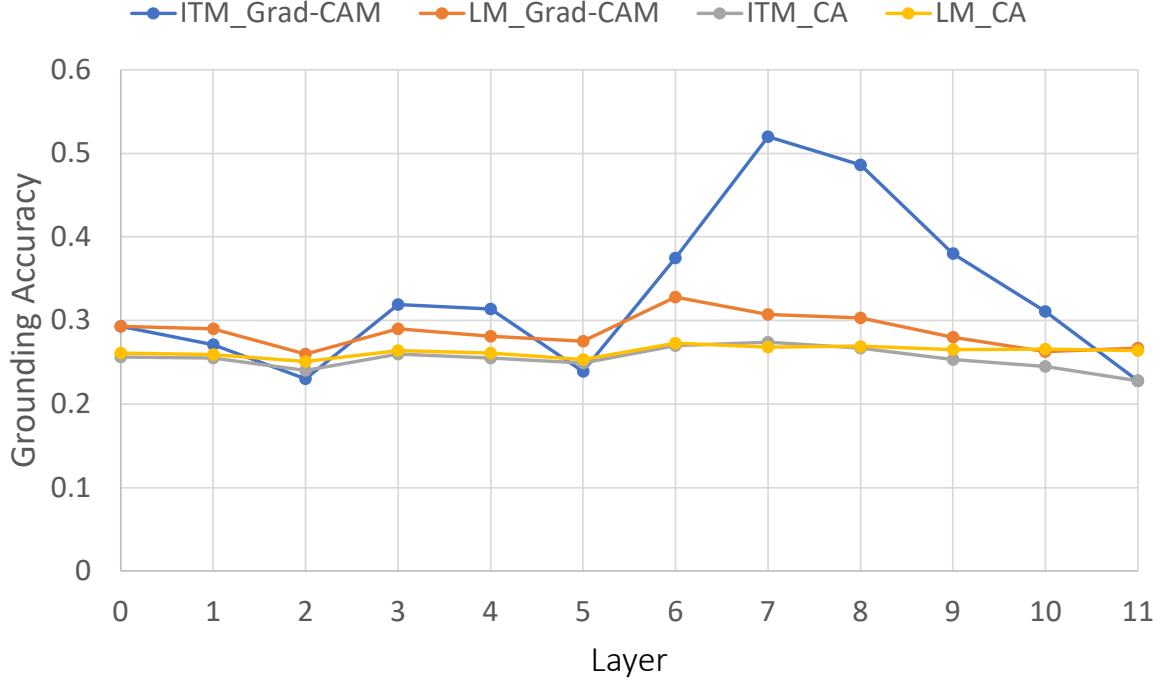


Figure B.12: Layer-wise comparative analysis of 4 association methods: Grad-CAM (gradient and cross-attention multiplication), CA (cross-attention only). ITM and LM represent using the corresponding parts of BLIP. The best performance is from the Grad-CAM of layer 7 from ITM.

1. For each image-caption pair, we decompose the image into $p \times p$ patches and the caption into t tokens. The two parts are then fed into the VIT and ITM parts of BLIP, respectively.
2. From BLIP, we extract the cross attentions (A^0 to A^{11}) and gradients (G^0 to G^{11}) over all heads in the best-performing layer, i.e., layer 7 based on our evaluation in Section B.4.1. Both A and G are with the shape $p^2 \times t$ (see Figure 4.5 for the notations).
3. We compute the Grad-CAM of each head using the corresponding A^i and G^i , and obtain an aggregated Grad-CAM C averaged over all C^i from individual heads. C is also with the shape of $p^2 \times t$.
4. C is then reshaped into t Grad-CAM maps, each with a shape of $p \times p$ and representing the distribution of the Grad-CAM from the corresponding word to all image patches.
5. In the meantime, we also feed the original $w \times h$ image into the SAM model, and generate m segments. Each is represented as a binary mask with the shape of $w \times h$.

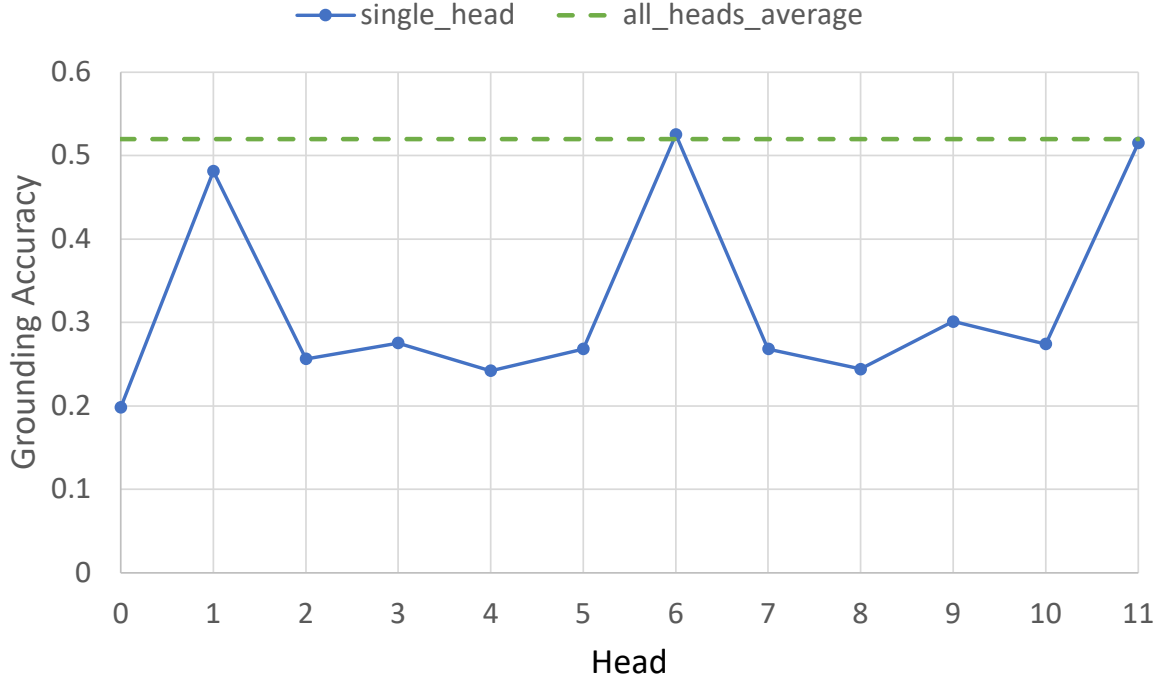


Figure B.13: Analysis of layer 7 heads in Grad-CAM: heads 1, 6, 11 excel in grounding accuracy, with the best-performing heads behave similarly to the Grad-CAM averaged over all heads.

6. Using the $p \times p$ Grad-CAM maps in Step 4 and the $w \times h$ segment masks in Step 5, we aggregate the Grad-CAM for each word-segment pair through the following steps:
 - (a) Resize the text token's Grad-CAM map from $p \times p$ to $w \times h$.
 - (b) Apply the corresponding segment mask (shape: $w \times h$) onto the resized Grad-CAM map (shape: $w \times h$), i.e., filter the Grad-CAM map to only contain values within the segment. The filtered map is called C_M .
 - (c) Compute the Grad-CAM score for this segment as $\Sigma C_M / \sqrt{\text{Area}(C_M)}$, where ΣC_M is the sum of values within the filtered Grad-CAM map, and $\text{Area}(C_M)$ is the number of pixels in the segment.

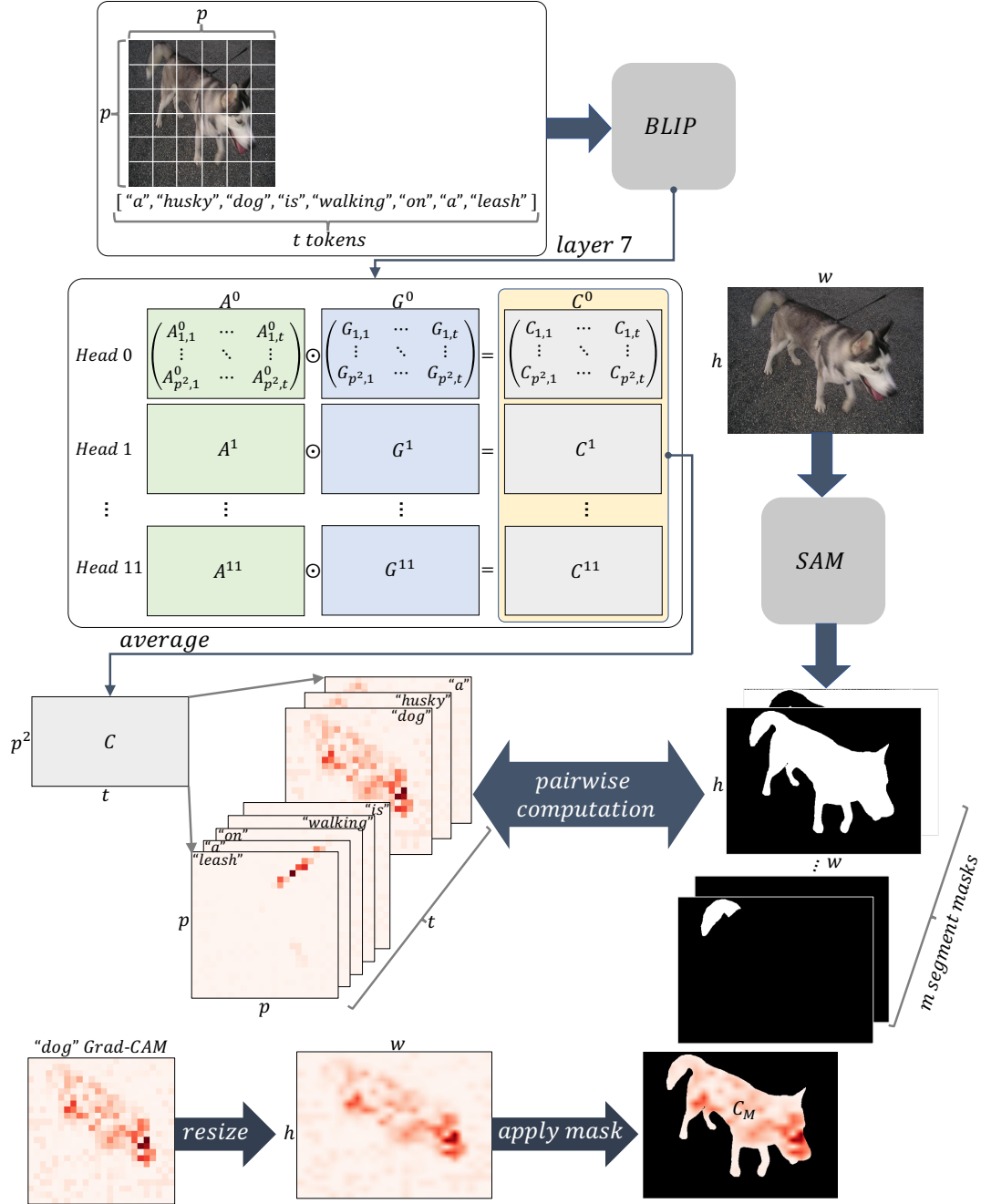


Figure B.14: Computation details of Grad-CAM aggregations.

REFERENCES

- [1] I. Kononenko, “Machine learning for medical diagnosis: history, state of the art and perspective,” *Artificial Intelligence in Medicine*, vol. 23, no. 1, pp. 89–109, 2001.
- [2] R. Najjar, “Redefining radiology: a review of artificial intelligence integration in medical imaging,” *Diagnostics*, vol. 13, no. 17, p. 2760, 2023.
- [3] T. Nguyen, J. Brandstetter, A. Kapoor, J. K. Gupta, and A. Grover, “Climax: A foundation model for weather and climate,” *arXiv preprint arXiv:2301.10343*, 2023.
- [4] V. Eyring, W. D. Collins, P. Gentine, E. A. Barnes, M. Barreiro, T. Beucler, M. Bocquet, C. S. Bretherton, H. M. Christensen, K. Dagon *et al.*, “Pushing the frontiers in climate modelling and analysis with machine learning,” *Nature Climate Change*, pp. 1–13, 2024.
- [5] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for youtube recommendations,” in *Proceedings of the 10th ACM Conference on Recommender Systems*, ser. RecSys ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 191–198.
- [6] H. Ko, S. Lee, Y. Park, and A. Choi, “A survey of recommendation systems: recommendation models, techniques, and application fields,” *Electronics*, vol. 11, no. 1, p. 141, 2022.
- [7] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, “A survey of autonomous driving: Common practices and emerging technologies,” *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.
- [8] D. Hendrycks, N. Carlini, J. Schulman, and J. Steinhardt, “Unsolved problems in ml safety,” *arXiv preprint arXiv:2109.13916*, 2021.
- [9] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, “Aligning ai with shared human values,” *arXiv preprint arXiv:2008.02275*, 2020.
- [10] Z. C. Lipton, “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery,” *Queue*, vol. 16, no. 3, pp. 31–57, 2018.
- [11] J. Thomas and K. Cook, “A visual analytics agenda,” *IEEE Computer Graphics and Applications*, vol. 26, no. 1, pp. 10–13, 2006.
- [12] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” *arXiv preprint arXiv:1702.08608*, 2017.
- [13] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, “Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1721–1730.
- [14] A. Adadi and M. Berrada, “Peeking inside the black-box: A survey on explainable artificial intelligence (XAI),” *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.
- [15] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees*. Taylor & Francis, 1984.

- [16] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, “Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model,” *The Annals of Applied Statistics*, vol. 9, no. 3, pp. 1350–1371, 2015.
- [17] M. T. Ribeiro, S. Singh, and C. Guestrin, “”Why should i trust you?”: Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.
- [18] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4765–4774.
- [19] S. Wachter, B. Mittelstadt, and C. Russell, “Counterfactual explanations without opening the black box: Automated decisions and the gdpr,” *Harvard Journal of Law & Technology*, vol. 31, p. 841, 2017.
- [20] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, “Visual analytics in deep learning: An interrogative survey for the next frontiers,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2674–2693, 2019.
- [21] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, “LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 667–676, 2018.
- [22] S. Liu, X. Wang, M. Liu, and J. Zhu, “Towards better analysis of machine learning models: A visual analytics perspective,” *Visual Informatics*, vol. 1, no. 1, pp. 48 – 56, 2017.
- [23] T. Spinner, U. Schlegel, H. Schäfer, and M. El-Assady, “explainer: A visual analytics framework for interactive and explainable machine learning,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1064–1074, 2019.
- [24] D. Sacha, M. Kraus, D. A. Keim, and M. Chen, “Vis4ml: An ontology for visual analytics assisted machine learning,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 385–395, 2019.
- [25] K. P. F.R.S., “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.
- [26] L. van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: <http://jmlr.org/papers/v9/vandermaaten08a.html>
- [27] P. E. Rauber, S. G. Fadel, A. X. Falcão, and A. C. Telea, “Visualizing the hidden activity of artificial neural networks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 101–110, 2017.
- [28] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, “Towards better analysis of deep convolutional neural networks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 91–100, 2017.

- [29] X. Zhao, Y. Wu, D. L. Lee, and W. Cui, “iForest: Interpreting random forests via visual analytics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 407–416, 2018.
- [30] T. Muhlbacher, L. Linhardt, T. Moller, and H. Piringer, “TreePOD: Sensitivity-aware selection of pareto-optimal decision trees,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 01, pp. 174–183, Jan. 2018.
- [31] D. Collaris, L. M. Vink, and J. J. van Wijk, “Instance-level explanations for fraud detection: A case study,” *CoRR*, vol. abs/1806.07129, 2018.
- [32] S. Liu, J. Xiao, J. Liu, X. Wang, J. Wu, and J. Zhu, “Visual diagnosis of tree boosting methods,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 163–173, 2017.
- [33] J. Wang, W. Zhang, L. Wang, and H. Yang, “Investigating the evolution of tree boosting models with visual analytics,” in *2021 IEEE 14th Pacific visualization symposium (PacificVis)*. IEEE, 2021, pp. 186–195.
- [34] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams, “Squares: Supporting interactive performance analysis for multiclass classifiers,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 61–70, 2016.
- [35] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh, “Modeltracker: Redesigning performance analysis tools for machine learning,” in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 337–346.
- [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] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.
- [38] Y. Ma, T. Xie, J. Li, and R. Maciejewski, “Explaining vulnerabilities to adversarial machine learning through visual analytics,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1075–1085, 2019.
- [39] M. Liu, S. Liu, H. Su, K. Cao, and J. Zhu, “Analyzing the noise robustness of deep neural networks,” in *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 2018, pp. 60–71.
- [40] 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)*. IEEE, 2020, pp. 271–275.
- [41] S. van den Elzen and J. J. van Wijk, “Baobabview: Interactive construction and analysis of decision trees,” in *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2011, pp. 151–160.

- [42] Y. Ming, P. Xu, F. Cheng, H. Qu, and L. Ren, "Protosteer: Steering deep sequence model with prototypes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 238–248, 2020.
- [43] L. Gou, L. Zou, N. Li, M. Hofmann, A. K. Shekar, A. Wendt, and L. Ren, "Vatld: a visual analytics system to assess, understand and improve traffic light detection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 261–271, 2020.
- [44] M. El-Assady, R. Kehlbeck, C. Collins, D. Keim, and O. Deussen, "Semantic concept spaces: Guided topic model refinement using word-embedding projections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1001–1011, 2020.
- [45] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89.
- [46] United States Environmental Protection Agency, "Drinking water distribution systems," <https://www.epa.gov/dwsixyearreview/drinking-water-distribution-systems>, 2019, accessed: 2021-06-17.
- [47] L. W. Mays, *Water distribution systems handbook*. McGraw-Hill, 2000. [Online]. Available: <https://www.accessengineeringlibrary.com/content/book/9780071342131/chapter/chapter7>
- [48] United States Environmental Protection Agency, "EPANET: Application for modeling drinking water distribution systems," <https://www.epa.gov/water-research/epanet>, 2020, accessed: 2021-07-17.
- [49] H. Mala-Jetmarova, N. Sultanova, and D. Savic, "Lost in optimisation of water distribution systems? A literature review of system operation," *Environmental Modelling & Software*, vol. 93, pp. 209–254, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1364815216307769>
- [50] J. April, F. Glover, J. P. Kelly, and M. Laguna, "Practical introduction to simulation optimization," in *Proceedings of the Winter Simulation Conference*. Citeseer, 2003, pp. 71–78.
- [51] O. Awe, S. Okolie, and O. Fayomi, "Optimization of water distribution systems: A review," *Journal of Physics: Conference Series*, vol. 1378, p. 022068, 2019.
- [52] F. Pecci, E. Abraham, and I. Stoianov, "Mathematical programming methods for pressure management in water distribution systems," *Procedia Engineering*, vol. 119, pp. 937–946, 2015.
- [53] T. C. Mosetlhe, Y. Hamam, S. Du, and E. Monacelli, "A survey of pressure control approaches in water supply systems," *Water*, vol. 12, no. 6, 2020.
- [54] J. E. van Zyl, D. A. Savic, and G. A. Walters, "Operational optimization of water distribution systems using a hybrid genetic algorithm," *Journal of Water Resources Planning and Management*, vol. 130, no. 2, pp. 160–170, 2004.

- [55] H. R. Maier, Z. Kapelan, J. Kasprzyk, J. Kollat, L. S. Matott, M. C. Cunha, G. C. Dandy, M. S. Gibbs, E. Keedwell, A. Marchi, A. Ostfeld, D. Savic, D. P. Solomatine, J. A. Vrugt, A. C. Zecchin, B. S. Minsker, E. J. Barbour, G. Kuczera, F. Pasha, A. Castelletti, M. Giuliani, and P. M. Reed, “Evolutionary algorithms and other metaheuristics in water resources: Current status, research challenges and future directions,” *Environmental Modelling and Software*, vol. 62, pp. 271–299, 12 2014.
- [56] J. Wang, S. Hazarika, C. Li, and H.-W. Shen, “Visualization and visual analysis of ensemble data: A survey,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 9, pp. 2853–2872, 2019.
- [57] K. Potter, A. Wilson, P.-T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson, “Ensemble-Vis: A framework for the statistical visualization of ensemble data,” in *Proc. ICDM*, 2009, pp. 233–240.
- [58] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead, “Noodles: A tool for visualization of numerical weather model ensemble uncertainty,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1421–1430, 2010.
- [59] L. Hao, C. G. Healey, and S. A. Bass, “Effective visualization of temporal ensembles,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 787–796, 2016.
- [60] M. Hummel, H. Obermaier, C. Garth, and K. I. Joy, “Comparative visual analysis of lagrangian transport in CFD ensembles,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2743–2752, 2013.
- [61] J. Waser, R. Fuchs, H. Ribicic, B. Schindler, G. Bloschl, and E. Groller, “World lines,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1458–1467, Nov. 2010.
- [62] L. van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: <http://jmlr.org/papers/v9/vandermaaten08a.html>
- [63] J. Knittel, A. Lalama, S. Koch, and T. Ertl, “Visual neural decomposition to explain multivariate data sets,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1374–1384, 2021.
- [64] Y. Wang, J. Li, F. Nie, H. Theisel, M. Gong, and D. J. Lehmann, “Linear discriminative star coordinates for exploring class and cluster separation of high dimensional data,” *Computer Graphics Forum*, vol. 36, no. 3, pp. 401–410, 2017.
- [65] T. Fujiwara, O.-H. Kwon, and K.-L. Ma, “Supporting analysis of dimensionality reduction results with contrastive learning,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 45–55, 2020.
- [66] X. Zhang, T. Fujiwara, S. Chandrasegaran, M. P. Brundage, T. Sexton, A. Dima, and K.-L. Ma, “A visual analytics approach for the diagnosis of heterogeneous and multidimensional machine maintenance data,” in *Proc. PacificVis*, 2021, pp. 196–205.

- [67] A. B. W. Kennedy and H. R. Sankey, “The thermal efficiency of steam engines. report of the committee appointed to the council upon the subject of the definition of a standard or standards of thermal efficiency for steam engines: With an introductory note. (including appendixes and plate at back of volume).” *Minutes of the Proceedings of the Institution of Civil Engineers*, vol. 134, no. 1898, pp. 278–312, 1898.
- [68] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen, “Characterizing provenance in visualization and data analysis: An organizational framework of provenance types and purposes,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 31–40, 2016.
- [69] M. C. Fu, “Optimization via simulation: A review,” *Annals of Operations Research*, vol. 53, no. 1, pp. 199–247, 1994.
- [70] Y. Carson and A. Maria, “Simulation optimization: Methods and applications,” in *Proc. WSC*, 1997, pp. 118—126.
- [71] J. Xu, E. Huang, C.-H. Chen, and L. H. Lee, “Simulation optimization: A review and exploration in the new era of cloud computing and big data,” *Asia-Pacific Journal of Operational Research*, vol. 32, no. 03, p. 1550019, 2015.
- [72] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
- [73] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 2019, pp. 4171–4186.
- [74] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized BERT pretraining approach,” *CoRR*, 2019.
- [75] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, “Temporal fusion transformers for interpretable multi-horizon time series forecasting,” *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.
- [76] 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 *9th International Conference on Learning Representations (ICLR)*, 2021.
- [77] P. Michel, O. Levy, and G. Neubig, “Are sixteen heads really better than one?” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [78] Y. Hao, L. Dong, F. Wei, and K. Xu, “Self-attention attribution: Interpreting information interactions inside transformer,” in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, 2021, pp. 12 963–12 971.

- [79] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky, “Revealing the dark secrets of BERT,” in *EMNLP-IJCNLP*, 2019, pp. 4365–4374.
- [80] C. Park, I. Na, Y. Jo, S. Shin, J. Yoo, B. C. Kwon, J. Zhao, H. Noh, Y. Lee, and J. Choo, “SANVis: Visual analytics for understanding self-attention networks,” in *2019 IEEE Visualization Conference (VIS)*, 2019, pp. 146–150.
- [81] J. F. DeRose, J. Wang, and M. Berger, “Attention Flows: Analyzing and comparing attention mechanisms in language models,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1160–1170, 2021.
- [82] R. Li, W. Xiao, L. Wang, H. Jang, and G. Carenini, “T3-vis: visual analytic for training and fine-tuning transformers in NLP,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2021, pp. 220–230.
- [83] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, “Towards better analysis of deep convolutional neural networks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 91–100, 2017.
- [84] J. Wang, L. Gou, H. Yang, and H.-W. Shen, “GANViz: A visual analytics approach to understand the adversarial game,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 6, pp. 1905–1917, 2018.
- [85] J. Wang, L. Gou, H.-W. 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, 2018.
- [86] Z. Jin, Y. Wang, Q. Wang, Y. Ming, T. Ma, and H. Qu, “GNNLens: A visual analytics approach for prediction error diagnosis of graph neural networks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 6, pp. 3024–3038, 2023.
- [87] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, “Visual analytics in deep learning: An interrogative survey for the next frontiers,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2674–2693, 2019.
- [88] J. Choo and S. Liu, “Visual analytics for explainable deep learning,” *IEEE Computer Graphics and Applications*, vol. 38, no. 4, pp. 84–92, 2018.
- [89] J. Vig, “A multiscale visualization of attention in the transformer model,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2019, pp. 37–42.
- [90] T. Jaunet, C. Kervadec, R. Vuillemot, G. Antipov, M. Baccouche, and C. Wolf, “VisQA: X-raying vision and language reasoning in transformers,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 976–986, 2022.
- [91] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

- [92] Z. Dong, T. Wu, S. Song, and M. Zhang, “Interactive attention model explorer for natural language processing tasks with unbalanced data sizes,” in *2020 IEEE Pacific Visualization Symposium*, 2020, pp. 46–50.
- [93] H. Strobel, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush, “Seq2seq-Vis: A visual debugging tool for sequence-to-sequence models,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 353–363, 2019.
- [94] E. Aflalo, M. Du, S.-Y. Tseng, Y. Liu, C. Wu, N. Duan, and V. Lal, “VI-interpret: An interactive visualization tool for interpreting vision-language transformers,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 21 406–21 415.
- [95] S. Abnar and W. Zuidema, “Quantifying attention flow in transformers,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Jul. 2020, pp. 4190–4197.
- [96] S. Jin, H. Lee, C. Park, H. Chu, Y. Tae, J. Choo, and S. Ko, “A visual analytics system for improving attention-based traffic forecasting models,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 1102–1112, 2023.
- [97] J. Cao, Z. Gan, Y. Cheng, L. Yu, Y.-C. Chen, and J. Liu, “Behind the scene: Revealing the secrets of pre-trained vision-and-language models,” in *European Conference on Computer Vision*, 2020, pp. 565–580.
- [98] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, “Do vision transformers see like convolutional neural networks?” in *NeurIPS*, vol. 34, 2021, pp. 12 116–12 128.
- [99] A. Bau, Y. Belinkov, H. Sajjad, N. Durrani, F. Dalvi, and J. Glass, “Identifying and controlling important neurons in neural machine translation,” in *International Conference on Learning Representations*, 2019.
- [100] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [101] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, “Transformers in vision: A survey,” *ACM Computing Surveys*, 2021.
- [102] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim *et al.*, “Explainable machine-learning predictions for the prevention of hypoxaemia during surgery,” *Nature Biomedical Engineering*, vol. 2, no. 10, pp. 749–760, 2018.
- [103] J. L. Koyner, K. A. Carey, D. P. Edelson, and M. M. Churpek, “The development of a machine learning inpatient acute kidney injury prediction model,” *Critical Care Medicine*, vol. 46, no. 7, p. 1070–1077, July 2018.

- [104] E. H. Shortliffe and M. J. Sepúlveda, “Clinical decision support in the era of artificial intelligence clinical decision support in the era of artificial intelligence clinical decision support in the era of artificial intelligence,” *JAMA*, vol. 320, no. 21, pp. 2199–2200, 12 2018.
- [105] A. Palczewska, J. Palczewski, R. Marchese Robinson, and D. Neagu, *Interpreting random forest classification models using a feature contribution method*. Cham: Springer, 2014, pp. 193–218.
- [106] E. Kawaler, A. Cobian, P. Peissig, D. Cross, S. Yale, and M. Craven, “Learning to predict post-hospitalization VTE risk from EHR data,” in *AMIA Annual Symposium Proceedings*, vol. 2012. American Medical Informatics Association, p. 436.
- [107] T. Zheng, W. Xie, L. Xu, X. He, Y. Zhang, M. You, G. Yang, and Y. Chen, “A machine learning-based framework to identify type 2 diabetes through electronic health records,” *International Journal of Medical Informatics*, vol. 97, pp. 120–127, 2017.
- [108] Tin Kam Ho, “Random decision forests,” in *Proceedings of International Conference on Document Analysis and Recognition*, vol. 1, Aug 1995, pp. 278–282 vol.1.
- [109] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [110] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins, and N. Khovanova, “Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation,” *Biomedical Signal Processing and Control*, 2017.
- [111] A. S. Goldfarb-Rumyantzev, J. D. Scandling, L. Pappas, R. J. Smout, and S. Horn, “Prediction of 3-yr cadaveric graft survival based on pre-transplant variables in a large national dataset,” *Clinical Transplantation*, vol. 17, no. 6, pp. 485–497, 2003.
- [112] M. Fouad, M. Mohamoud, M. Hagag, and A. Akl, “Prediction of long term living donor kidney graft outcome: Comparison between rule based, decision tree and linear regression,” *International Journal of Advanced Research in Computer Science*, vol. 3, p. 185, 04 2015.
- [113] Y. A. Reshef, D. N. Reshef, H. K. Finucane, P. C. Sabeti, and M. Mitzenmacher, “Measuring dependence powerfully and equitably,” *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 7406–7468, 2016.
- [114] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
- [115] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, “LightGBM: A highly efficient gradient boosting decision tree,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.
- [116] A. V. Dorogush, V. Ershov, and A. Gulin, “CatBoost: gradient boosting with categorical features support,” *CoRR*, vol. abs/1810.11363, 2018.

- [117] Z. C. Lipton, D. C. Kale, C. Elkan, and R. C. Wetzel, “Learning to diagnose with LSTM recurrent neural networks,” in *Proceedings of the 4th International Conference on Learning Representations*, 2016.
- [118] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, “Recurrent neural networks for multivariate time series with missing values,” *Scientific Reports (Nature Publisher Group)*, vol. 8, pp. 1–12, 04 2018.
- [119] A. Saabas, “TreeInterpreter,” <https://github.com/andosa/treeinterpreter>, accessed: 2019-12-19.
- [120] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, “RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 3504–3512.
- [121] B. C. Kwon, M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo, “RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 299–309, 2019.
- [122] L. Devroye and T. Wagner, “Distribution-free performance bounds for potential function rules,” *IEEE Transactions on Information Theory*, vol. 25, no. 5, pp. 601–604, Sep 1979.
- [123] L. Devroye, “Exponential inequalities in nonparametric estimation,” in *Nonparametric Functional Estimation and Related Topics*. Springer, 1991, pp. 31–44.
- [124] G. Lugosi and M. Pawlak, “On the posterior-probability estimate of the error rate of nonparametric classification rules,” *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 475–481, Mar 1994.
- [125] M. Kearns and D. Ron, “Algorithmic stability and sanity-check bounds for leave-one-out cross-validation,” *Neural Computation*, vol. 11, no. 6, pp. 1427–1453, Aug 1999.
- [126] O. Bousquet and A. Elisseeff, “Stability and generalization,” *Journal of Machine Learning Research*, vol. 2, no. Mar, pp. 499–526, 2002.
- [127] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*. MIT Press, 1994.
- [128] D. Haussler, “Part 1: Overview of the probably approximately correct (PAC) learning framework,” <http://web.cs.iastate.edu/~honavar/pac.pdf>, 1995, accessed: 2019-12-19.
- [129] K. Pearson and F. Galton, “Vii. Note on regression and inheritance in the case of two parents,” *Proceedings of the Royal Society of London*, vol. 58, no. 347-352, pp. 240–242, 1895.
- [130] C. Spearman, “The proof and measurement of association between two things,” *The American Journal of Psychology*, vol. 100, no. 3/4, pp. 441–471, 1987.

- [131] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. New York, NY, USA: Wiley-Interscience, 2006.
- [132] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, “Detecting novel associations in large data sets,” *Science*, vol. 334, no. 6062, pp. 1518–1524, 2011.
- [133] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, “MIMIC-III, a freely accessible critical care database,” *Scientific Data*, vol. 3, p. 160035, 2016.
- [134] D. N. Reshef, Y. A. Reshef, P. C. Sabeti, and M. Mitzenmacher, “An empirical study of the maximal and total information coefficients and leading measures of dependence,” *The Annals of Applied Statistics*, vol. 12, no. 1, pp. 123–155, 03 2018.
- [135] S. Romano, N. X. Vinh, K. Verspoor, and J. Bailey, “The randomized information coefficient: Assessing dependencies in noisy data,” *Machine Learning*, vol. 107, no. 3, pp. 509–549, 2018.
- [136] P. G. Policar, M. Strazar, and B. Zupan, “openTSNE: a modular python library for t-SNE dimensionality reduction and embedding,” *BioRxiv*, p. 731877, 2019.
- [137] W. S. Cleveland, “Robust locally weighted regression and smoothing scatterplots,” *Journal of American Statistical Association*, vol. 74, no. 368, pp. 829–836, 1979.
- [138] T. Fujiwara, O.-H. Kwon, and K.-L. Ma, “Supporting analysis of dimensionality reduction results with contrastive learning,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 45–55, 2020.
- [139] M. Bakator and D. Radosav, “Deep learning and medical diagnosis: A review of literature,” *Multimodal Technologies and Interaction*, vol. 2, no. 3, 2018.
- [140] X. Mei, H.-C. Lee, K.-y. Diao, M. Huang, B. Lin, C. Liu, Z. Xie, Y. Ma, P. M. Robson, M. Chung, A. Bernheim, V. Mani, C. Calcagno, K. Li, S. Li, H. Shan, J. Lv, T. Zhao, J. Xia, Q. Long, S. Steinberger, A. Jacobi, T. Deyer, M. Luksza, F. Liu, B. P. Little, Z. A. Fayad, and Y. Yang, “Artificial intelligence-enabled rapid diagnosis of patients with COVID-19,” *Nature Medicine*, vol. 26, no. 8, pp. 1224–1228, 2020.
- [141] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *Advances in Neural Information Processing Systems*, vol. 28, pp. 91–99, 2015.
- [142] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, “A survey of deep learning techniques for autonomous driving,” *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.
- [143] V. Tyrsa, A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy, “A review of deep learning methods and applications for unmanned aerial vehicles,” *Journal of Sensors*, vol. 2017, p. 3296874, 2017.

- [144] I. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
- [145] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, “Robust physical-world attacks on deep learning visual classification,” in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [146] N. Akhtar and A. Mian, “Threat of adversarial attacks on deep learning in computer vision: A survey,” *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.
- [147] 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.
- [148] F. Hohman, H. Park, C. Robinson, and D. H. P. 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, 2019.
- [149] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks,” *Advances in Neural Information Processing Systems*, vol. 29, pp. 3387–3395, 2016.
- [150] C. Olah, A. Mordvintsev, and L. Schubert, “Feature visualization,” *Distill*, 2017, <https://distill.pub/2017/feature-visualization>.
- [151] J. L. Long, N. Zhang, and T. Darrell, “Do convnets learn correspondence?” in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.
- [152] Y. Ma and R. Maciejewski, “Visual analysis of class separations with locally linear segments,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 1, pp. 241–253, 2020.
- [153] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *arXiv preprint arXiv:1706.06083*, 2017.
- [154] B. Zhou, A. Khosla, Àgata Lapedriza, A. Oliva, and A. Torralba, “Object detectors emerge in deep scene CNNs,” in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
- [155] A. Mordvintsev, C. Olah, and M. Tyka, “DeepDream - a code example for visualizing neural networks,” *Google Research*, vol. 2, no. 5, 2015. [Online]. Available: <http://googleresearch.blogspot.co.uk/2015/07/deepdream-code-example-for-visualizing.html>
- [156] J. Alonso and Y. Chen, “Receptive field,” *Scholarpedia*, vol. 4, no. 1, p. 5393, 2009.
- [157] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.

- [158] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.
- [159] D. Müllner, “Modern hierarchical, agglomerative clustering algorithms,” *arXiv:1109.2378*, 2011.
- [160] A. Krizhevsky, “Learning multiple layers of features from tiny images,” *Master’s thesis, University of Toronto*, 2009.
- [161] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2018.
- [162] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [163] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *CoRR*, vol. abs/1605.07146, 2016. [Online]. Available: <http://arxiv.org/abs/1605.07146>
- [164] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, “Adversarial examples are not bugs, they are features,” in *Advances in Neural Information Processing Systems*, H. Wallach and H. Larochelle and A. Beygelzimer and F. d’Alché-Buc and E. Fox and R. Garnett, Ed., vol. 32. Curran Associates, Inc., 2019.
- [165] J. Li, R. Selvaraju, A. Gotmare, S. Joty, C. Xiong, and S. C. H. Hoi, “Align before fuse: Vision and language representation learning with momentum distillation,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 9694–9705, 2021.
- [166] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.
- [167] J. Li, D. Li, C. Xiong, and S. Hoi, “Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 12 888–12 900.
- [168] M. T. Ribeiro, S. Singh, and C. Guestrin, “” why should i trust you?” explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.
- [169] D. Bertucci, M. M. Hamid, Y. Anand, A. Ruangrotsakun, D. Tabatabai, M. Perez, and M. Kahng, “Dendromap: Visual exploration of large-scale image datasets for machine learning with treemaps,” *IEEE Transactions on Visualization and Computer Graphics (IEEE VIS 2022 Conference)*, 2022. [Online]. Available: <https://div-lab.github.io/dendromap/>
- [170] S. Kaul, D. Borland, N. Cao, and D. Gotz, “Improving visualization interpretation using counterfactuals,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 998–1008, 2021.

- [171] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.
- [172] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, “Cider: Consensus-based image description evaluation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4566–4575.
- [173] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [174] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” *arXiv preprint arXiv:1405.0312*, 2014.
- [175] E. Strickland, “Andrew ng, ai minimalist: The machine-learning pioneer says small is the new big,” *IEEE Spectrum*, vol. 59, no. 4, pp. 22–50, 2022.
- [176] J. Wang, S. Liu, and W. Zhang, “Visual analytics for machine learning: A data perspective survey,” *arXiv preprint arXiv:2307.07712*, 2023.
- [177] 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, 2018.
- [178] X. Zhang, J. P. Ono, H. Song, L. Gou, K.-L. Ma, and L. Ren, “Sliceteller: A data slice-driven approach for machine learning model validation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 842–852, 2022.
- [179] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, “Embedding projector: Interactive visualization and interpretation of embeddings,” *arXiv preprint arXiv:1611.05469*, 2016.
- [180] A. Karpathy, “t-sne visualization of cnn codes,” <https://cs.stanford.edu/people/karpathy/cnnembed/>, accessed: 2023-09-19.
- [181] A. Bäuerle, C. v. Onzenoodt, D. Jönsson, and T. Ropinski, “Semantic Hierarchical Exploration of Large Image Datasets,” in *EuroVis 2023 - Short Papers*, T. Hoellt, W. Aigner, and B. Wang, Eds. The Eurographics Association, 2023.
- [182] X. Xie, X. Cai, J. Zhou, N. Cao, and Y. Wu, “A semantic-based method for visualizing large image collections,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 7, pp. 2362–2377, 2019.
- [183] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu, “A survey of visual analytics techniques for machine learning,” *Computational Visual Media*, vol. 7, pp. 3–36, 2021.
- [184] 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, 2020.

- [185] W. Yang, X. Wang, J. Lu, W. Dou, and S. Liu, “Interactive steering of hierarchical clustering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 10, pp. 3953–3967, 2020.
- [186] Y. Ming, P. Xu, F. Cheng, H. Qu, and L. Ren, “Protosteer: Steering deep sequence model with prototypes,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 238–248, 2019.
- [187] H. Strobel, J. Kinley, R. Krueger, J. Beyer, H. Pfister, and A. M. Rush, “Genni: Human-ai collaboration for data-backed text generation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 1106–1116, 2021.
- [188] W. Yang, X. Ye, X. Zhang, L. Xiao, J. Xia, Z. Wang, J. Zhu, H. Pfister, and S. Liu, “Diagnosing ensemble few-shot classifiers,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 9, pp. 3292–3306, 2022.
- [189] S. Jia, Z. Li, N. Chen, and J. Zhang, “Towards visual explainable active learning for zero-shot classification,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 791–801, 2021.
- [190] C. Chen, J. Wu, X. Wang, S. Xiang, S.-H. Zhang, Q. Tang, and S. Liu, “Towards better caption supervision for object detection,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 4, pp. 1941–1954, 2021.
- [191] S. Brade, B. Wang, M. Sousa, S. Oore, and T. Grossman, “Promptify: Text-to-image generation through interactive prompt exploration with large language models,” in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023, pp. 1–14.
- [192] Y. Feng, X. Wang, K. K. Wong, S. Wang, Y. Lu, M. Zhu, B. Wang, and W. Chen, “Prompt-Magician: Interactive prompt engineering for text-to-image creation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 295–305, 2024.
- [193] “GPT-4 technical report,” 2024. [Online]. Available: <https://arxiv.org/abs/2303.08774>
- [194] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *arXiv preprint arXiv:2307.09288*, 2023.
- [195] C. Yeh, Y. Chen, A. Wu, C. Chen, F. Viégas, and M. Wattenberg, “Attentionviz: A global view of transformer attention,” *arXiv preprint arXiv:2305.03210*, 2023.
- [196] Z. J. Wang, R. Turko, and D. H. Chau, “Dodrio: Exploring Transformer Models with Interactive Visualization,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2021, pp. 132–141.
- [197] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, “Segment anything,” *arXiv preprint arXiv:2304.02643*, 2023.

- [198] A. Vedaldi and S. Soatto, “Quick shift and kernel methods for mode seeking,” in *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12–18, 2008, Proceedings, Part IV 10*. Springer, 2008, pp. 705–718.
- [199] Z. Zhao, P. Xu, C. Scheidegger, and L. Ren, “Human-in-the-loop extraction of interpretable concepts in deep learning models,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 780–790, 2021.
- [200] “Textblob: Simplified text processing,” <https://textblob.readthedocs.io/>, accessed: 2023-09-19.
- [201] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”, 2009.
- [202] M. Ghoniem, J.-D. Fekete, and P. Castagliola, “A comparison of the readability of graphs using node-link and matrix-based representations,” in *IEEE Symposium on Information Visualization*, 2004, pp. 17–24.
- [203] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 618–626.
- [204] D. Li, J. Li, H. Le, G. Wang, S. Savarese, and S. C. Hoi, “LAVIS: A one-stop library for language-vision intelligence,” in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 31–41. [Online]. Available: <https://aclanthology.org/2023.acl-demo.3>
- [205] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg, “Referitgame: Referring to objects in photographs of natural scenes,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 787–798.
- [206] A. Krizhevsky *et al.*, “Learning multiple layers of features from tiny images,” 2009.