INTERACTIVE SCALABLE DISCOVERY OF CONCEPTS, EVOLUTIONS, AND VULNERABILITIES IN DEEP LEARNING

A Dissertation
Presented to
The Academic Faculty

By

Haekyu Park

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering

Georgia Institute of Technology

December 2023

© Haekyu Park 2023

INTERACTIVE SCALABLE DISCOVERY OF CONCEPTS, EVOLUTIONS, AND VULNERABILITIES IN DEEP LEARNING

Thesis committee:

Duen Horng (Polo) Chau, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Judy Hoffman
School of Interactive Computing
Georgia Institute of Technology

Cong (Callie) Hao
School of Electrical and Computer Engineering
Georgia Institute of Technology

Diyi Yang
Computer Science Department
Stanford University

Chao Zhang
School of Computational Science and Engineering
Georgia Institute of Technology

Date approved: October 11, 2023

It is not the strongest of the species that survive, nor the most intelligent,

but the one most responsive to change.

Charles Darwin

To my parents, for being my role models

ACKNOWLEDGMENTS

Over the past five years, I have been fortunate to learn from many amazing people. This experience has been more than just about gaining knowledge; it's been a journey of personal growth, and understanding complex aspects of life. I am thankful to everyone who has contributed to this enriching experience.

Most importantly, I thank my family—my parents, sister, and brother, for their constant love, which have been key to my resilience and success. They have always been a source of comfort and motivation.

My advisor, Polo, deserves special recognition for his invaluable guidance. His expertise and insight have not only shaped my research but also my personal and professional growth. He has been more than a mentor to me, teaching me about important life qualities.

I'm also grateful for the opportunity to work with the brilliant people at the Polo Club of Data Science at Georgia Tech. I want to thank my academic siblings Fred Hohman, Nilaksh Das, Rahul Duggal, Scott Freitas, Jay Wang, Austin Wright, Seongmin Lee, Anthony Peng, Shang-Tse Chen, and Minsuk Kahng for their our time spent together to develop exciting research ideas.

My internships at NVIDIA RAPIDS, Microsoft Research, and Stripe were crucial in growing me as a machine learning researcher. My mentors Dr. Bartley Richardson, Dr. Brad Rees, Dr. Joe Eaton, Dr. Gonzalo Ramos, and Revanth Rameshkumar, thank you all for being incredibly great mentors.

I am also thankful to the members of my thesis committee, Dr. Polo Chau, Dr. Judy Hoffman, Dr. Callie Hao, Dr. Diyi Yang, and Dr. Chao Zhang, for providing your invaluable feedback in shaping my dissertation.

I also want to thank my wonderful friends in Atlanta who have helped me navigate through my PhD, ensuring that I never felt alone and always had happy moments together: Jiyeon Kim, Dongjin Choi, Allison Koo, Haejun Han, EJ Yun, and Dongsuk Sung. Additionally, I am forever grateful to my lifelong best friends from Korea: Juhyun Park, Minkung Kim, Sumi Hwang, Heejae Rim, Sungwon Yoo, and Jaehee Lee, who have always been a constant and comforting presence in my life.

A special thanks to Eric Choi, the love of my life, for his never-ending support, care, and the peace he brings to my heart. Thank you for always being with me.

Finally, my sincere thanks go to all who have played a part in my journey. Your influence has been vital, and I am grateful to each one of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Deep Neural Networks (DNNs) are increasingly prevalent, but deciphering their operations is challenging. Such a lack of clarity undermines trust and problem-solving during deployment, highlighting the urgent need for interpretability. How can we efficiently summarize concepts models learn? How do these concepts evolve during training? When models are at risk from potential threats, how do we explain their vulnerabilities?

We address these concerns with a human-centered approach, by developing novel systems to interpret learned concepts, their evolution, and potential vulnerabilities within deep learning. This thesis focuses on three key thrusts:

(1) **Scalable Automatic Visual Summarization of Concepts.** We develop NEUROCARTOGRAPHY, an interactive system that scalably summarizes and visualizes concepts learned by a large-scale DNN, such as InceptionV1 trained with 1.2M images. A large-scale human evaluation with 244 participants shows that NEUROCARTOGRAPHY discovers coherent, human-meaningful concepts.

(2) **Insights to Reveal Model Vulnerabilities.** We develop scalable interpretation techniques to visualize and identify internal elements in DNNs, which are susceptible to potential harms, aiming to understand how these defects lead to incorrect predictions. We develop first-of-its-kind interactive systems such as BLUFF that visually compares the activation pathways for benign and attacked images in DNNs, and SKELETONVIS that explains how attacks manipulate human joint detection in human action recognition models.

(3) **Scalable Discovery of Concept Evolution During Training.** Our first-of-its-kind CONCEPTEVO unified interpretation framework holistically reveals the inception and evolution of learned concepts and their relationships during training. CONCEPTEVO enables powerful new ways to monitor model training and discover training issues, addressing critical limitations of existing post-training interpretation research. A large-scale human evaluation with 260 participants demonstrates that CONCEPTEVO identifies concept evolutions that are both meaningful to humans and important for class predictions.

This thesis contributes to information visualization, deep learning, and crucially, their intersection. We have developed open-source interactive interfaces, scalable algorithms, and a unified framework for interpreting DNNs across different models. Our work impacts academia, industry, and the government. For example, our work has contributed to the DARPA GARD program (Garanteeing AI Robustness against Deception). Additionally, our work has been recognized through a J.P. Morgan AI PhD Fellowship and 2022 Rising Stars in IEEE EECS. NEUROCARTOGRAPHY has been highlighted as a top visualization publication (top 1%) invited to SIGGRAPH.

CHAPTER 1
INTRODUCTION

Deep Neural Networks (DNNs) have demonstrated remarkable success across various applications, including human action recognition for video understanding [1, 2], intelligent image retrieval and captioning [3, 4, 5], and machine vision for medical imaging [6, 7].

Despite their successes, understanding how DNNs work and what they have learned remains a fundamental challenge. The black box nature of DNNs, stemming from their vast parameter space, hinders people from auditing and trusting their decision-making processes. When models underperform or fall victim to malicious attacks, there is a lack of actionable guidance for understanding vulnerabilities and implementing effective fixes. Conventional approaches such as hyperparameter tuning, while enhancing model performance, offer limited insights into the causes of underperformance and directions for vulnerability improvements [8]. Moreover, existing interpretation approaches predominantly center around post-training analysis [9, 10], leaving a significant gap in understanding the evolution of models during the training process. This chasm includes a lack of insight into training deficiencies such as poor generalizability [11, 12, 13] and convergence failures [14, 15], which can potentially result in wasting time and resources [16, 17], if the training ultimately fails to achieve desired outcomes.

The challenges outlined above have sparked a call to action for **interpreting deep learning**, prompting pivotal questions: How can we scalably discover and summarize concepts learned within DNNs? How do we identify and explain vulnerabilities in DNNs? How does the evolution of learned concepts unfold throughout DNNs' training process? This thesis introduces novel algorithms and visual interactive techniques designed to address these challenges.

1.1 Thesis Goal: Motivation and Vision

Through my extensive research across diverse fields such as machine learning interpretability, defense mechanisms against adversarial attacks on DNNs, and information visualization over the past five years, I have come to an important insight: the key to unlocking the full potential of DNNs lies in human-centered interpretation.

DNNs excel when augmenting human capabilities, but their full empowerment is contingent upon clear interpretability. Without transparency, people may struggle to trust,

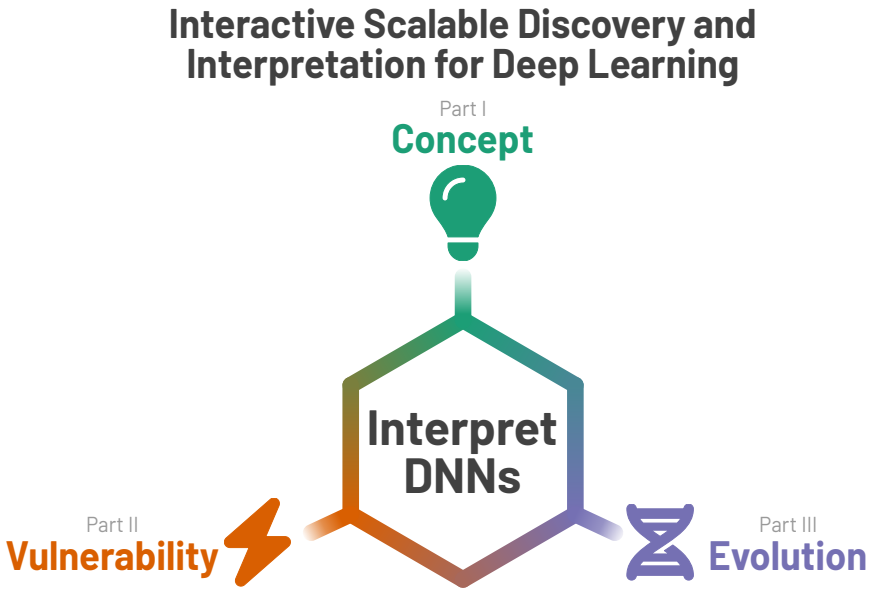


Figure 1.1: An overview of my interdisciplinary research where I design and develop scalable interactive discovery of **concepts**, **vulnerabilities**, and **evolutions** in deep learning.

employ, or amend these models effectively. Given that humans are the primary users of DNNs, aligning interpretability with human cognition is essential. This fundamental belief drives the overarching goal of my thesis: to bridge the gap between complex DNN architectures and intuitive human understanding, through a comprehensive human-centered interpretation of the inner workings, vulnerabilities, and evolutions of DNNs.

1.2 Thesis Overview

To enable human-centered DNN interpretability on their inner workings, vulnerabilities, and evolutions, this thesis focuses on answering three complementary research questions: how to summarize concepts learned in DNNs (Part I); how to explain their vulnerabilities (Part II); and how to reveal concept evolutions in training process (Part III). Considering the immense complexity of DNN parameters, scalability is an integral part of my thesis. Similarly, to enrich the interpretation process from a human perspective, my thesis integrates carefully designed interactive elements and visualization techniques. Fig 1.2 summarizes how the three complementary research questions are answered by the corresponding parts of this thesis and example works.

1.2.1 Part I: Scalable Automatic Visual Summarization of Concepts

Deep Neural Networks (DNNs) have become ubiquitous across various domains; however, their intricate architecture and numerous parameters make their interpretation notably chal-

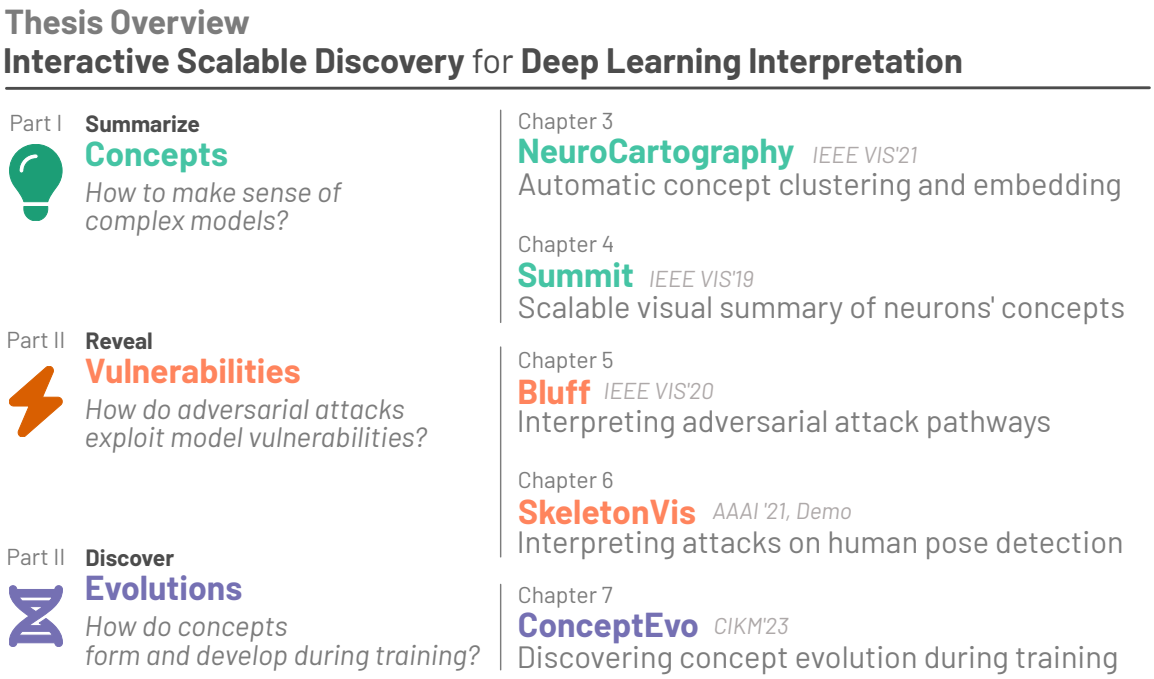


Figure 1.2: This thesis consists of three parts. Each part is represented by one block with its research question and example works that map to the chapters of the thesis.

lenging. Without effective interpretations, people might struggle to comprehend and trust the results generated by these models. As a response, this part focuses on interpreting a large-scale DNN's inner workings, by visually summarizing the fundamental concepts it learns and how these concepts contribute to the model's behavior.

NEUROCARTOGRAPHY*: Scalable Automatic Visual Summarization of Concepts in Deep Neural Networks (Chapter 3)*

We introduce NEUROCARTOGRAPHY, an interactive system that scalably summarizes and visualizes concepts learned by neural networks (Fig 1.3). It automatically discovers and groups neurons that detect the same concepts, and describes how such neuron groups interact to form higher-level concepts and the subsequent predictions NEUROCARTOGRAPHY introduces two scalable summarization techniques: (1) *neuron clustering* groups neurons based on the semantic similarity of the concepts detected by neurons (e.g., neurons detecting "dog faces" of different breeds are grouped); and (2) *neuron embedding* encodes the associations between related concepts based on how often they co-occur (e.g., neurons detecting "dog face" and "dog tail" are placed closer in the embedding space). Through a large-scale human evaluation, we demonstrate that our technique discovers neuron groups that represent coherent, human-meaningful concepts. And through usage scenarios, we describe how our approaches enable interesting and surprising discoveries, such as concept cascades of related concepts or the existence of isolated concepts.

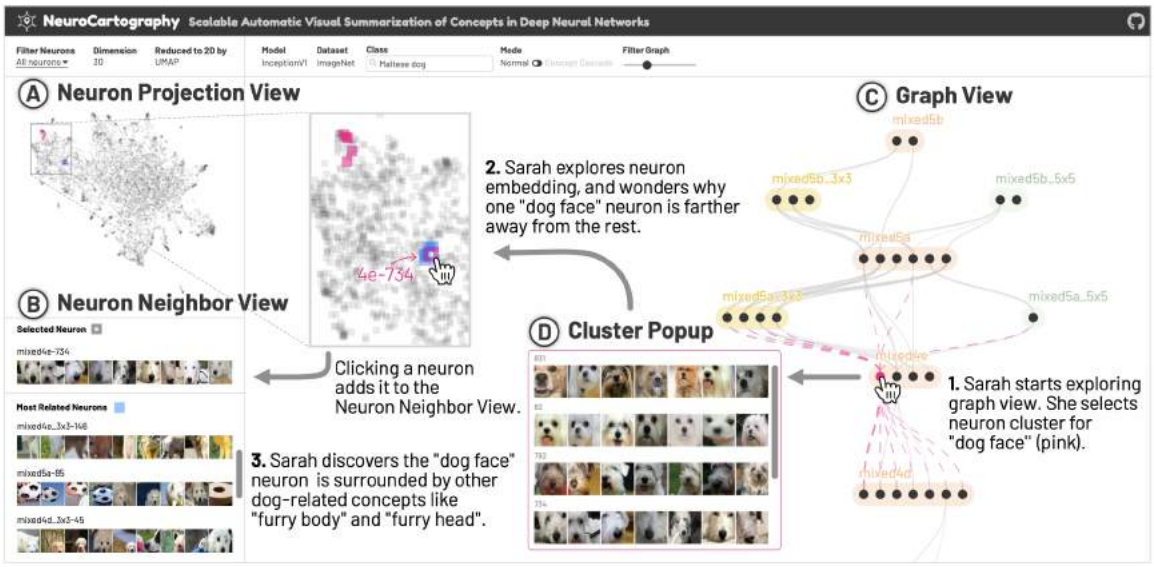


Figure 1.3: NEUROCARTOGRAPHY scalably summarizes concepts learned in a DNN, by automatically discovering and visualizing groups of neurons detecting similar concepts. **1.** Within Graph View (at C), users can explore the interaction of neuron clusters—represented as nodes in the graph—that collaboratively influence a specific class prediction, such as the *Maltese dog* class. Clicking on a cluster (circle) visualizes the concepts detected by its member neurons, paired with corresponding dataset examples (at D). **2.** For the clicked cluster, its constituent neurons are highlighted in Neuron Projection View (at A), which spatially organizes neurons by related concepts, positioning similar concepts in closer proximity. **3.** In Neuron Projection View, when users select a specific neuron, such as one for "dog face," Neuron Neighbor View (at B) shows the selected neuron's closely related ones and their concepts, like neurons for "furry body" or "furry head."

SUMMIT*: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations (Chapter 4)*

NEUROCARTOGRAPHY builds on foundations of research my earlier PhD research SUMMIT, the first interactive system that scalably visualizes what features each neuron has learned and how those features interact to make predictions. NEUROCARTOGRAPHY extends these ideas to automatically cluster and embed the learned concepts. SUMMIT introduces two scalable summarization techniques: (1) *activation aggregation* discovers important neurons, and (2) *neuron-influence aggregation* identifies relationships among such neurons. SUMMIT combines these techniques to create the novel *attribution graph* that reveals and summarizes crucial neuron associations and substructures that contribute to a model's outcomes. SUMMIT scales to large data, such as the ImageNet dataset with 1.2M images, and leverages neural network feature visualization and dataset examples to help users distill large, complex neural network models into compact, interactive visualizations. The SUMMIT visualization runs in modern web browsers and is open-sourced.

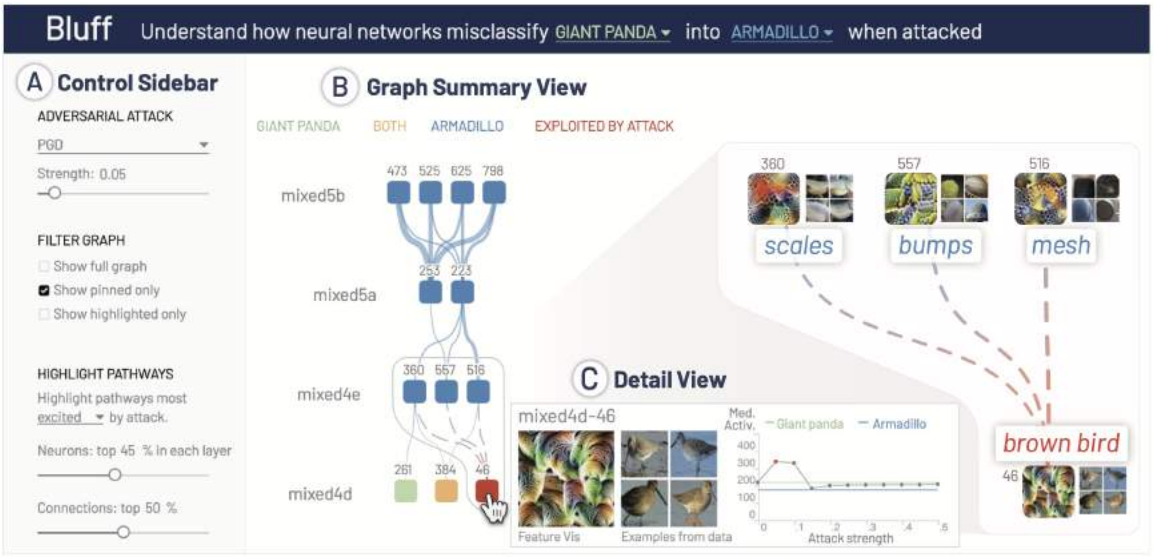


Figure 1.4: BLUFF enables interactive visualization of how adversarial attacks infiltrate DNNs that cause incorrect outcomes. For example, it clarifies why an InceptionV1 misclassifies adversarial giant panda images, that are crafted by the Projected Gradient Descent (PGD) attack, as armadillo. PGD perturbed pixels to induce "brown bird" feature, an appearance more likely shared by an armadillo (small, roundish, brown body) than a panda, activating more features that contribute to the armadillo (mis)classification (e.g., "scales," "bumps," "mesh"). The adversarial pathways, formed by these neurons and their connections, overwhelm the benign panda pathways and lead to the ultimate misclassification.

1.2.2 Part II: Insights to Reveal Model Vulnerabilities

In the preceding part, we focus on interpreting the decision-making process of a normally performing DNN. However, DNNs are not infallible; they can occasionally be susceptible to potential harms such as adversarial attacks. Small, human-imperceptible noise injected into inputs can easily fool DNNs into making wrong predictions, raising alarms for safety-critical applications, such as autonomous driving and data-driven healthcare [18, 19, 20, 21, 22, 23]. How can we identify and interpret the vulnerabilities inherent in DNNs? To address this concern, we have developed first-of-its-kind interactive systems:

1. BLUFF that visualizes and compares the activation pathways for benign and attacked images in vision-based DNNs.
2. SKELETONVIS that visualizes and explains how attacks manipulate human joints in human action recognition models.

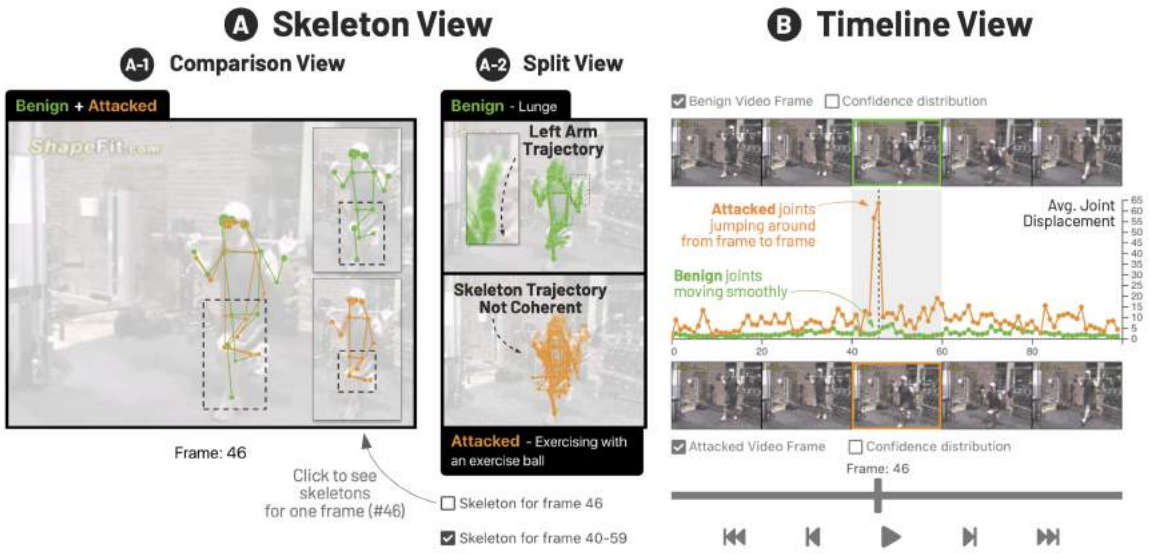


Figure 1.5: The interface of SKELETONVIS, visualizing how the *Fast Gradient Method* attack manipulates the left foot joints detected by the Detectron2 Keypoint R-CNN model. (A) The Skeleton View shows the joints perturbed to unexpected locations. (B) Timeline View reveals the attacked joints spuriously jumping around from one frame to the next, leading to a "spike" in the average joint displacement across attacked frames. These manipulations finally sway the ST-GCN action detection model into misclassifying the attacked frames as "exercising with exercise ball," instead of the correct "lunge" classification.

BLUFF*: Interactively Deciphering Adversarial Attacks on Neural Networks (Chapter 5)*

BLUFF is an interactive visualization tool for demystifying adversarial attacks on vision-based DNNs. It visualizes how attacks *inhibit* neurons detecting concepts essential for the benign class and *excite* those driving misclassification (Fig 1.4). At its core, BLUFF visually compares activation pathways—comprising neurons and their interrelations showing strong activation to input images—resulting from both benign and adversarial inputs. BLUFF pinpoints where a model is exploited by an attack, how they are used, and what impact the exploitation has on the final prediction, across multiple attack strengths, revealing various strategies that adversarial attacks employ to inflict harm on a model. By offering side-by-side visual comparisons of these pathways, users can readily discern the divergence points where adversarial pathways deviate from their corresponding benign pathways, resulting in an incorrect prediction.

SKELETONVIS*: Interactive Visualization for Understanding Adversarial Attacks on Human Action Recognition Models (Chapter 6)*

With BLUFF, we acquire a deeper understanding about the impact of adversarial attacks on image classifiers. We extend to the domain of human-action recognition, driven by its

widespread real-world applications such as home robotics, healthcare for the aging population, and surveillance. We present SKELETONVIS, an interactive system that enhances understanding the impacts of adversarial attack on human pose detection models, visualizing manipulated joint detections (Fig 1.5). Furthermore, SKELETONVIS provides quantitative measurements across video frames to capture the abnormal signals present in the attacked frames, helping users easily identify the specific frames exploited by the attacks.

1.2.3 Part III: Scalable Discovery of Concept Evolution During Training

The previous two parts primarily focus on interpreting fully trained DNNs in terms of their learned concepts and vulnerabilities. That is, such interpretation occurs after the training process. Moreover, many existing approaches predominantly center around post-training interpretation [10, 9], offering limited insights into how models evolve as they are trained, even though interpreting the model evolution has emerged as a promising direction to monitor the network training [9, 8]. Crucially, a gap exists in our understanding of how a model progresses during training and its association with model deficiencies like poor generalizability [13, 11, 12] or convergence failures [15, 14]. Relying solely on post-training interpretation poses challenges for real-time discovery and diagnosis during training, potentially resulting in the inefficient use of time and resources [16, 17], particularly if the training ultimately fails to achieve desired outcomes. How can we help people interpret the dynamic evolution of a model as it is trained?

CONCEPTEVO*: Interpreting Concept Evolution in DNN Training (Chapter 7)*

We develop CONCEPTEVO, a first-of-its-kind general unified interpretation framework of DNNs that reveals the inception and evolution of learned concepts during training (Fig 1.6). CONCEPTEVO introduces two novel technical contributions: (1) an algorithm that generates a unified semantic space, enabling side-by-side comparison of different models during training, and (2) an algorithm that discovers and quantifies important concept evolutions for class predictions. It aids in uncovering potential issues during model training and provides insights into their causes, such as: (1) severely harmed concept diversity caused by incompatible hyperparameters (e.g., overly high learning rate) as shown in Fig 1.6b; and (2) slowly evolving concepts despite rapid increases in training accuracy in overfitted model as shown in Fig 1.6c. A large-scale human evaluation with 260 participants demonstrates that CONCEPTEVO identifies concept evolutions that are both meaningful to humans and important for class predictions.

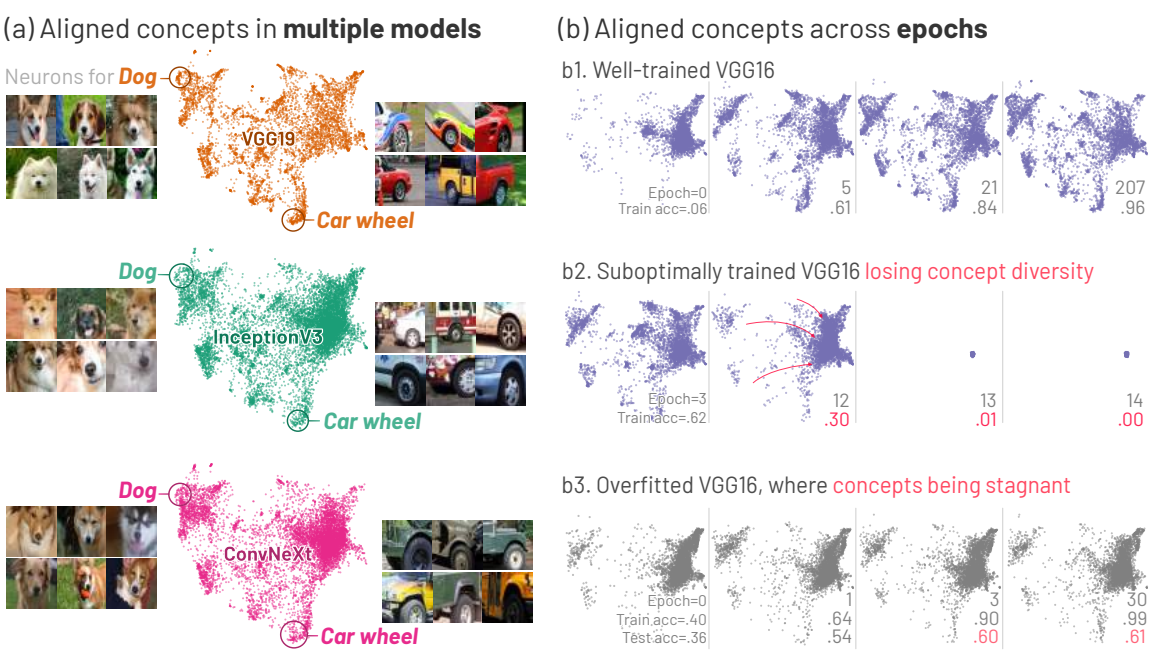


Figure 1.6: CONCEPTEVO creates a unified semantic space that enables side-by-side comparison of concepts learned by different models during training. Within this space, neurons (represented as dots) that detect similar concepts (such as a dog face or a car wheel) are embedded and aligned at similar location. This layout makes it straightforward to visualize the inception and evolution of these concepts across different DNN models during training. Additionally, CONCEPTEVO serves as a diagnostic tool for identifying potential issues in model training. For example, in (b2), if a model starts losing the diversity of its learned concepts, CONCEPTEVO can flag this as a concern. To illustrate, (b1) visualizes the evolution of a well-trained VGG16 that shows gradual concept formations and refinements. In (b2), a VGG16 suboptimally trained with a large learning rate rapidly loses the ability to detect most concepts. In (b3), a overfitted VGG16 without dropout layers shows slow concept evolutions despite rapid training accuracy increases. We abbreviate "top-5 training/test accuracies" as "train/test acc."

1.3 Thesis Statement

By adopting human-centered approaches, we can enhance our understanding of the intrinsically complex DNNs, by:

1. summarizing learned **concepts** through scalable automatic algorithms and visualizations,
2. revealing **vulnerabilities** exploited by adversarial attacks, and
3. discovering **evolutions** of concepts during training.

1.4 Research Contributions

This thesis makes research contributions to multiple fields, including interactive data visualization, machine learning, and more importantly their intersection, by interpreting DNNs' learned concepts (Part I), vulnerabilities (Part II), and evolutions (Part III).

New scalable systems for global, unified model interpretation.

- NEUROCARTOGRAPHY introduces two innovative scalable concept summarization techniques, *neuron clustering* and *neuron embedding*, to **automatically represent the vast conceptual space of all neurons in large DNNs and their relationships**, offering a holistic interpretation of large datasets like ImageNet with 1.2M images. By avoiding exhaustive comparisons between neuron pairs' concepts, our methods achieve linear time complexity with respect to the number of neurons, far surpassing the conventional quadratic time. A large-scale human evaluation with 244 participants shows that these concept summarization techniques discover coherent, human-meaningful concepts (Chapter 3).
- CONCEPTEVO is **the first unified framework for interpreting model training of multiple DNNs**, by refining our neuron embedding algorithm to synchronize the conceptual spaces from different models across training epochs into a unified semantic space. This alignment not only streamlines the comparative analysis of different model training processes, but also provides a powerful means to monitor model training and detects training anomalies. Our extensive human evaluation, involving 260 participants, demonstrates that our framework identifies concept evolutions that are both meaningful to humans and important for class predictions (Chapter 7).
- We develop BLUFF and SKELETONVIS, each a **first-of-its kind interactive system for deciphering attacks** to DNNs. BLUFF visualizes and compares the activation pathways for benign and attacked images in vision-based neural networks (Chapter 5). SKELETONVIS visualizes and explains how attacks manipulate human joints in human action recognition models (Chapter 6).

Surprising discoveries and new insights.

- BLUFF sheds light on the previously unknown vulnerabilities of DNNs. Notably, we are the **first to visualize and characterize the variation in attack strategies based on attack intensity**. For instance, milder attacks may adopt a 'death by a thousand cuts' approach, while stronger attacks focus on exploiting a select few neurons. These discoveries can guide the creation of robust defensive measures, such as the elimination of susceptible neurons from the model. (Chapter 5)

- NEUROCARTOGRAPHY is among the **first system to map the comprehensive concept landscape of large DNNs, revealing unexpected phenomena**, such as the ex-

istence of isolated concept—a unique feature not tied to any particular image characteristics (e.g., "watermark" concept can appear on almost any kinds of images). Such surprising insights help guide model compression, suggesting the potential removal of such isolated neuron clusters. (Chapter 3)

- CONCEPTEVO **empowers users with new ways to identify potential model training issues** such as: (1) incompatible hyperparameters (e.g., overly high learning rate) severely harm concept diversity; and (2) concepts in overfitted models evolve slowly despite rapid training accuracy increases. Leveraging these insights allows for timely interventions, like halting training early when concept diversity is at risk. (Chapter 7)

Democratizing access to interpretability research through open-source systems.

- Prioritizing user convenience, our interactive visual systems, such as NEUROCARTOGRAPHY (Chapter 3), BLUFF (Chapter 5), and SKELETONVIS (Chapter 6), are presented as **interactive web applications**. These tools are universally accessible, compatible with any modern web browser, and free from the hassles of additional installations or the constraints of specific hardware.

- To foster a collaborative environment for DNN interpretability, we have made the **source codes of our algorithms available to the public**, including NEUROCARTOGRAPHY (Chapter 3), BLUFF (Chapter 5), and CONCEPTEVO (Chapter 7).

1.5 Impact

This thesis is making impact to academia, industry, and the government.

- NEUROCARTOGRAPHY (Chapter 3) has been highlighted as a top visualization publication (top 1% of 442 submissions) invited to present at SIGGRAPH.

- BLUFF (Chapter 5), SKELETONVIS, NEUROCARTOGRAPHY (Chapter 3), and CONCEPTEVO (Chapter 7) have been contributing to the multi-million dollar DARPA GARD (Guaranteeing AI Robustness against Deception) program in understanding model robustness and devising effective defenses.

- This dissertation has been recognized by a 2021 J.P. Morgan AI PhD Fellowship, among the only 15 awards in the world. Additionally, it was highlighted in 2022 Rising Stars in EECS, an international academic career workshop focusing on electrical engineering, computer science, and artificial intelligence.

- Research ideas developed in this dissertation contributed to multiple high-impact projects aimed at broadening the access to high-quality machine learning education.

Among them are the NVIDIA Data Science Teaching Kit [24] now freely available to thousands of educators around the world, and CNN Explainer [25] that has been integrated into academic programs at institutions worldwide, such as Georgia Tech (Deep Learning), University of Wisconsin-Madison (Intro to AI), and University of Kyoto (Bioengineering).

1.6 Prior Publications and Authorship

The research in this thesis results from years of collaboration with my advisor, Duen Horng (Polo) Chau, and colleagues at Georgia Institute of Technology. To honor their contributions, the first-person plural perspective will be used throughout. Relevant publications are detailed in Table 1.1.

Part I: Scalable Automatic Visual Summarization of Concepts	
Chapter 3:	NeuroCartography: Scalable Automatic Visual Summarization of Concepts in Deep Neural Networks. Haekyu Park, Nilaksh Das, Rahul Duggal, Austin P. Wright, Omar Shaikh, Fred Hohman, Duen Horng Chau. *IEEE Visualization Conference (VIS)*, 2021.
Chapter 4:	Summit: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations. Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng (Polo) Chau. *IEEE Visualization Conference (VIS)*, 2019.
Part II: Insights to Reveal Model Vulnerabilities	
Chapter 5:	Bluff: Interactively Deciphering Adversarial Attacks on Deep Neural Networks. Nilaksh Das*, Haekyu Park*, Zijie J. Wang, Fred Hohman, Robert Firstman, Emily Rogers, Duen Horng Chau. (*Authors contributed equally) *IEEE Visualization Conference (VIS)*, 2020.
Chapter 6:	SkeletonVis: Interactive Visualization for Understanding Adversarial Attacks on Human Action Recognition Models. Haekyu Park, Zijie J. Wang, Nilaksh Das, Anindya S. Paul, Pruthvi Perumalla, Zhiyan Zhou, Duen Horng Chau. *The AAAI Conference on Artificial Intelligence, Demo*, 2021.
Part III: Scalable Discovery of Concept Evolution During Training	
Chapter 7:	Concept Evolution in Deep Learning Training: A Unified Interpretation Framework and Discoveries. Haekyu Park, Seongmin Lee, Benjamin Hoover, Austin Wright, Omar Shaikh, Rahul Duggal, Nilaksh Das, Kevin Li, Judy Hoffman, Duen Horng (Polo) Chau. *International Conference on Information and Knowledge Management*, 2023.

Table 1.1: The publications mapped to the thesis outline

CHAPTER 2
BACKGROUND AND RELATED WORK

A DNN takes as input a data instance and outputs its prediction (e.g., class label). Transforming the input to the output involves billions of numerical computations, the scale of which is incomprehensible to humans. Recent research aims to summarize these calculations into abstract human-understandable representation. In this chapter, we provide an overview of existing research for neural network interpretability that motivates our work.

2.1 Interpreting Well-Performing Fully-trained DNNs

2.1.1 DNN Interpretation Through Local Surrogate Explanation Models

An effective strategy for interpreting the predictions made by complex models is to adopt a simpler *explanation model* that serves a dual purpose: First, it should aptly approximate the behavior of the complex original model, at least around a specific instance (i.e., being locally faithful). Second, it should be simple enough to be readily understood by humans.

For example, Local Interpretable Model-agnostic Explanations (LIME) [26] is designed to create an interpretable explanation model, such as a linear regression or a decision tree, that closely mimics the original model's predictions for a particular instance and its neighboring data instances. Fig 2.1 presents an example of LIME explaining a Google's Inception neural network [27]. LIME first approximates the image classifier as a linear model that processes input images using super-pixels as features, with the most influential super-pixels corresponding to the highest coefficients in the linear model, where it can highlight the crucial super-pixels for each class prediction. For example, the first subfigure of Fig 2.1b shows that a super-pixel representing "fretboard" is important for the "Electric guitar" prediction, which looks quite natural to humans.

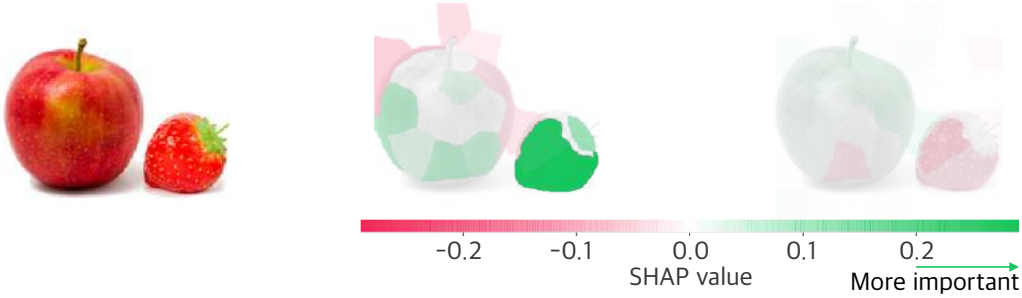
Similarly, SHapley Additive exPlanations (SHAP) [30] identifies a linear explanation model that can approximate the prediction for a specific input data instance. Such linear explanation model is intentionally crafted to compute importance scores for individual features, represented by the features' corresponding Shapely values [31, 32]. Fig 2.2 illustrates SHAP's explanation for a VGG16 model [33]. Much like LIME explanations do, SHAP explanations compute the importance score (i.e., Shapely value) for each segmented component of an input image.

Example of a Surrogate Explanation Approach: LIME

(a) Input image (b) LIME explanation for *Electric guitar, Acoustic guitar, and Labrador*

Figure 2.1: LIME simplifies neural network image classifier predictions into an easily understandable linear model. It uses super-pixels as input features, highlighting the most influential ones with higher coefficient weights. For example, for an input image (Fig 2.1a) classified as Electric guitar ($p = 0.32$), Acoustic guitar ($p = 0.24$), and Labrador ($p = 0.21$), LIME identifies most influential features (super-pixels) for each class prediction. It flags the super-pixel representing "fretboard" as important for the "Electric guitar" prediction (Fig 2.1b). The figures have been sourced from the LIME paper [26].

Example of a Surrogate Explanation Approach: SHAP



(a) Input image (b) SHAP explanation for *Strawberry* and *Granny Smith* (a type of apple)

Figure 2.2: SHAP interprets neural network image classifier predictions using a linear model that treats input images as collections of super pixels. It calculates the importance scores for each super pixel using Shapely values, with higher values indicating greater importance. For example, for an input image containing both an apple and strawberry (Fig 2.2a), SHAP identifies and highlights the relevant image segments for the "Strawberry" and "Granny Smith" (a type of apple) class predictions by made by a VGG16. The figures have been sourced the SHAP GitHub readme [28] and one of its example notebook [29].

Introducing surrogate explanation models comes with a notable limitation: it often yield inconsistent explanations, raising concerns about their reliability and robustness [34, 35, 36, 37, 38]. This inconsistency arises because surrogate explanations do not adequately consider the distribution of feature values, relying on random perturbations for generating (synthetic) contextual data points around a specific instance [34, 39]. Even minor local data variations can result in significant interpretation differences [37, 38]. Moreover, surrogate explanations tend to oversimplify complex original models into very simple ones,

which can significantly misrepresent highly nonlinear DNNs [40]. They are also vulnerable to various risks such as adversarial attacks and biases [35, 39]. This inconsistency and vulnerability stem from the surrogate model's divergence from the original network, potentially causing it to overlook crucial structural aspects in the original model [40, 41].

2.1.2 Direct Input Feature Attribution: Salience Methods

To address the vulnerability associated with surrogate explanations, a fundamental alternative is to gain a direct understanding of the network itself, bypassing any intermediary surrogate explanations. A viable approach is to evaluate how variations in input features influence the output directly through the network. This helps identify specific input features of greater importance, particularly those whose variations result in more significant output changes.

For example, salience methods [42, 43, 44] quantify the sensitivity of the output to each input pixel (feature) by computing the gradient of the loss function for a class with respect to the input pixels. This computation generates a saliency map, which shows the significance of input pixels for the class prediction, featuring both high negative and positive values. Additionally, Grad-CAM [45] computes the gradient from the loss function to the last convolutional layer (rather than all the way back to the image), generating a coarse localization map that highlights important regions within the image. As depicted in Fig 2.3, saliency maps visually exemplify the importance of individual pixels for image predictions. For example, they help clarify the recognition of a dog's face in the "Wheaten Terrier" image prediction.

However, these methods face a challenge, as crucial image pixels often lack meaningful or detailed information to comprehend the workings of the black box [46, 47, 48, 49]. For example, in the medical domain, interpreting what an saliency map precisely elucidates can be challenging due to potential ambiguity; a group of pixels might represent multiple overlapping shapes, textures, and landmarks [48, 50]. Merely localizing the region of an image does not reveal precisely what elements within that area the network deemed valuable [49, 50]. This inherent ambiguity introduces the risk of both misinterpretation and overinterpretation of the explanations [48, 49].

2.1.3 Concept-Based Interpretation by Looking into the Network

To overcome the limitation of salience methods, which often provide incomplete explanations that fail to convey why a deep learning model makes a specific prediction, recent studies have shifted their focus towards explaining high-level, human-understandable concepts learned within the deep neural networks and their relevance to the models' predictions. This deeper understanding is achieved through a thorough examination of the network's internal processes.

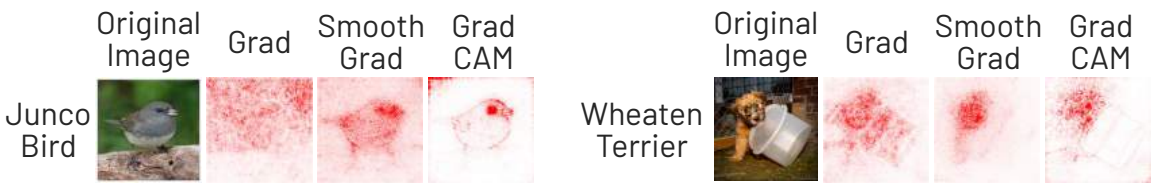


Figure 2.3: Saliency maps for two different inputs processed by an InceptionV3 model trained on ImageNet [51]. These maps are generated using various techniques: Grad (Gradient explanations) [42], SmoothGrad [43], and GradCam [45]. These saliency maps highlight the significance of individual pixels concerning image predictions (darker means more important), such as the recognition of a dog's face for the "Wheaten Terrier" image prediction. These figures have been sourced from [44].

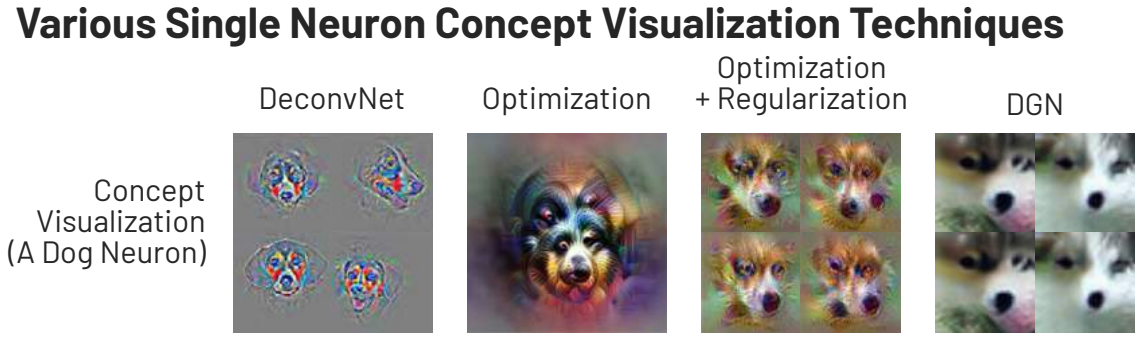


Figure 2.4: Visualization of a single neuron's detected concept (in this case, a neuron detecting a dog's face) created using various techniques such as DeconvNet [52], optimization [53], optimization with regularization [54], and Deep Generator Network (DGN) [55]. The figures have been sourced from [52, 56, 54, 55].

A common approach is to decode and visualize the concepts a single neuron has learned by identifying input patterns that strongly trigger the neuron's responses. For example, DeconvNet visualization reveals the input stimuli that excite individual neurons through a top-down projection: it takes a neuron's activation map (i.e., the activation of the layer to which the neuron belongs, with all other neurons' activation maps set to zero) and processes it through an approximate inverse network of the original model (i.e., DeconvNet) until reaching the input pixel space [52]. Feature visualization techniques generate synthesized images to maximize neuron activations, revealing concepts that strongly activate these neurons [56, 57]. For example, gradient-based approach [53] formulates an optimization problem to maximize neuron activations and subsequently performs gradient ascent in the input space to obtain the synthetic optimal image. However, such gradient methods can be vulnerable, tending to generate high-frequency patterns as adversarial examples that may not occur in real-life scenarios and may not represent what the neuron genuinely detects. Some approaches introduce regularization techniques to address this challenge [54].

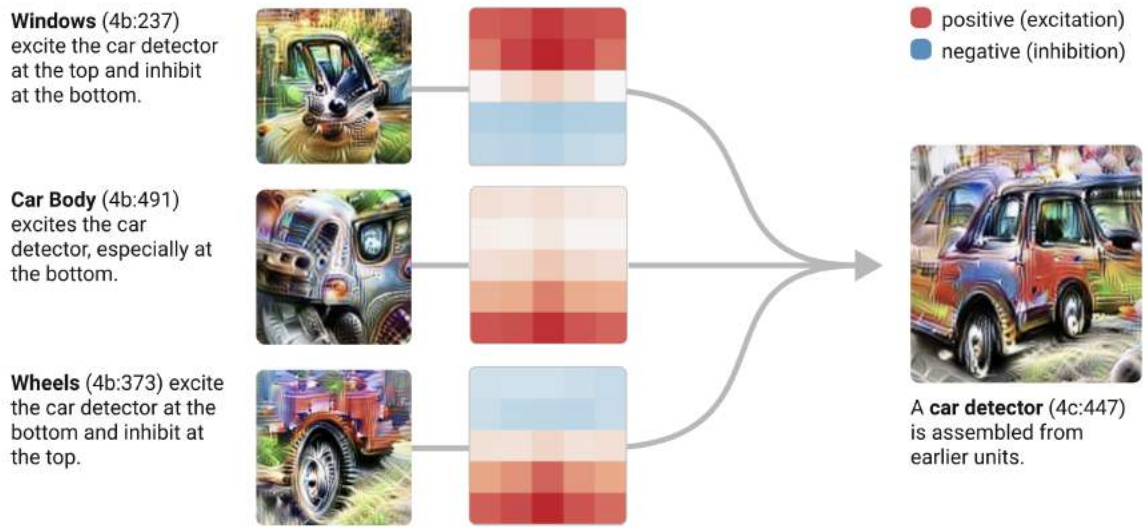


Figure 2.5: The circuits-study [58] explores the neural networks' sub-graphs, which are composed closely interconnected features connected by weights, forming circuits. Such circuits can be meticulously analyzed for a deeper understanding. For example, consider a neuron that detects cars (right). By using concepts from preceding layers (left) such as "windows," "car body," and "wheels," the neuron identifies windows at the upper part, due to the upper portion of its convolutional filter, and wheels at the bottom. The figures have been sourced from [58].

Another approach [55] trains a deep generator network to produce high-quality synthetic images that not only represent the concepts learned by each neuron but also appear realistic. Fig 2.4 illustrates examples of the visualization of "dog face" concept detected by a neuron, using these methods.

These techniques for visualizing single neuron concepts have provided compelling insights into how DNNs construct their internal hierarchical representations: earlier layers focus on low-level concepts (e.g., colors, textures), while higher layers handle high-level concepts [53]. Expanding upon these insights, various methods have been developed not only to identify the concepts of a neuron but also to reveal relationships between them. For example, the study on hierarchical modular representation maps out a tree-structured relationship among hidden layer units in models trained with MNIST digit data, explaining how lower-level features, such as simple lines and curves, hierarchically evolve into more intricate features, like combinations of multiple curves for the digit "3" [59]. Similarly, Circuits-study examines the connections between neurons, suggesting that concepts detected by neurons are interlinked by weights, thereby either exciting and restraining concepts among them [58], as illustrated in Fig 2.5.

Several studies have emerged to quantify concept-based interpretation. For example, TCAV utilizes vectorized activations in each layer, employing them in a binary classification task to determine the quantitative sensitivity of an interpretable concept (e.g., a striped

Figure 2.6: Adversarial examples that deceive AlexNet [63]: Injecting imperceptible noise (center) into an input image (left) produces an adversarial image (right) misclassified as *Ostrich*. The figures have been sourced from [64].

pattern) to a model's decision for a specific class prediction (e.g., zebra) [47]. Network Dissection [60] and Net2Vec [61] introduce methods for quantifying interpretability by measuring the alignment between filter activations and concepts. ACE [62] extends TCAV by automatically generating concepts, such as image segmentations, while TCAV typically requires a predefined set of concepts. ACE's automatic concept generation allows for both quantification and the discovery of important concepts influencing class predictions.

2.2 Interpreting Malfunctioning DNNs That Are under Attack

While DNNs have made remarkable strides across various applications, it is imperative to acknowledge their susceptibility to potential threats, particularly adversarial attacks. These attacks aim to confuse a given DNN model into making incorrect predictions by adding carefully crafted, but seemingly imperceptible perturbations to the input. For example, as depicted in Fig 2.6, when imperceptible, non-random perturbations are applied to an image, it becomes possible to change a DNN's prediction. The vulnerability of DNNs to the adversarial attacks raises significant concerns, especially in safety-critical domains such as autonomous driving and data-driven healthcare [19, 20, 21, 65].

Several studies have probed into the underlying causes of this vulnerability. One prominent hypothesis, as proposed in [19], posits that the inherent linearity within neural networks significantly contributes to their susceptibility to adversarial perturbations. Linear models may inadvertently assign excessive importance to perturbed signals, often at the expense of more significant signals. Zhang et al. [66] have developed a method to identify vulnerable neurons, particularly those whose activation intensity varies sensitively between benign and adversarial examples. Ilyas et al. [67] demonstrated that adversarial examples can be directly attributed to the presence of non-robust features. These features, derived from patterns in the data distribution, are highly predictive yet brittle, making them incomprehensible to humans. Furthermore, Zheng et al. [68] have delved into interpreting

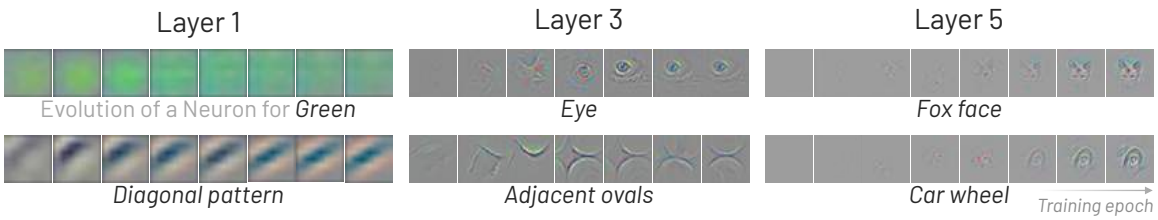


Figure 2.7: DeconvNet visualization [52] can track the evolution of neurons' detected concept during training. Here are examples of concept evolution for some neurons, observed at epochs 1, 2, 5, 10, 20, 30, 40, and 64. It appears that the lower layers of DNNs converge relatively quickly, within just a few epochs. In contrast, the upper layers take more time, solidifying only after a substantial number of epochs (between 40 to 50). This highlights the importance of allowing DNNs to train until they achieve full convergence. The figures have been sourced from the DeconvNet visualization paper [52].

the classification process of adversarial examples, revealing how these perturbations affect the prediction. Notably, their observations indicate that adversarial perturbations tend to influence latent variables gradually, layer by layer, rather than directly altering features in adversarial examples.

Several visual analytics approaches have been developed to visually interpret adversarial attacks on DNNs. AEVis [69] compares critical neurons and their connections for both benign and adversarial inputs, facilitating the investigation of adversarial examples' working mechanisms. Ma et al. [70] have created a visual analytics framework to explain and explore model vulnerabilities to adversarial attacks. This framework employs a multifaceted visualization scheme designed to support the analysis of data poisoning attacks from various perspectives, including models, data instances, features, and local structures.

2.3 Interpreting Evolution of DNNs During Training

Existing literature has predominantly emphasized the interpretation of fully trained DNNs, analyzing their learned concepts and vulnerabilities. Such interpretations typically occur after the model has completed its training, offering limited insights into the model's evolution during its training phase [10, 9]. This limitation persists despite the growing recognition of the importance of monitoring model evolution during training [9, 8].

This oversight leaves an unresolved gap: understanding a model's progression during training and its implications for potential deficiencies. These deficiencies can encompass issues such as inadequate generalizability [13, 11, 12] and convergence failures [15, 14]. Relying solely on post-training interpretation hampers real-time diagnosis during the training process. This can lead to inefficiencies, potentially squandering time and resources, especially if the desired outcomes remain unachieved [16, 17].

To bridge this critical gap, several studies have ventured into this domain, aiming to interpret DNNs *during* their training. These studies delve into the transformation of data representations within models across epochs, and the implications of these changes for performance [71, 72, 73]. DeconvNet-based concept visualization method [52] visualizes the evolution of concepts detected by neurons during training. As seen in Fig 2.7, it observed that lower layers of the model tend to converge within a few epochs, while the upper layers require a considerable number of epochs (typically 40-50) to fully develop, underscoring the importance of allowing models to train until convergence. DeepEyes [74] investigates the dynamic behavior of individual neurons for various classes during the training process. Similarly, DGMTracker [8] offers an analysis of weight, activation, and gradient shifts over training time. Other techniques provide a 2D projection of neuron evolution in relation to specific labels [75, 76], although this might narrow our comprehension to only the labels in view. DeepView [77] proposes metrics to determine the evolutionary diversity of neurons for classification tasks.

Part I

Scalable Automatic Visual Summarization of Concepts

OVERVIEW

While DNNs have become widely used across various domains, yet interpreting their inner workings still remains challenging. With limited tools for interpretation, people may struggle to effectively scrutinize and comprehend the rationale behind these models' decisions. Notably, there has been a push from both government and industry to enhance model interpretability [78, 79].

Responding to this need, Part I focuses on interpreting DNNs' decision-making logic, particularly through the visualization and summarization of important concepts learned within large-scale DNNs. Part I begins with NEUROCARTOGRAPHY (Chapter 3), an interactive system that scalably summarizes and visualizes concepts learned by neural networks. It automatically discovers and groups neurons that detect similar concepts, and describes how such neuron groups interact to form higher-level concepts and the subsequent predictions. Additionally, it encodes such concepts and their semantic relationships into vectors, offering a coherent visual map of the entire concepts a model has learned. Chapter 3 is adapted from work that was published at VIS 2021 [80].

> **Chapter 3**
> **NEUROCARTOGRAPHY: Scalable Automatic Visual Summarization of Concepts in Deep Neural Networks.** Haekyu Park, Nilaksh Das, Rahul Duggal, Austin P. Wright, Omar Shaikh, Fred Hohman, Duen Horng Chau. *IEEE Visualization Conference (VIS), 2021.*

NEUROCARTOGRAPHY is inspired by SUMMIT, which summarizes and visualizes what concepts individual neurons detect and how those concepts interact to make predictions. Chapter 4 is adapted from SUMMIT, that was published at VIS 2019 [78].

> **Chapter 4**
> **SUMMIT: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations.** Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng (Polo) Chau. *IEEE Visualization Conference (VIS), 2019.*

CHAPTER 3

NEUROCARTOGRAPHY: SCALABLE AUTOMATIC VISUAL SUMMARIZATION OF CONCEPTS IN DEEP NEURAL NETWORKS

Existing research on making sense of deep neural networks often focuses on interpreting single neurons. However, this approach often does not fully represent how multiple neurons collaboratively encode complex concepts. We present NEUROCARTOGRAPHY, an interactive system that scalably summarizes and visualizes concepts learned by neural networks. It automatically discovers and groups neurons that detect the same concepts, and describes how such neuron groups interact to form higher-level concepts and the subsequent predictions. NEUROCARTOGRAPHY introduces two scalable summarization techniques: (1) *neuron clustering* groups neurons based on the semantic similarity of the concepts detected by neurons (e.g., neurons detecting "dog faces" of different breeds are grouped); and (2) *neuron embedding* captures the associations between related concepts based on how often they co-occur (e.g., neurons detecting "dog face" and "dog tail" are placed closer in the embedding space). An essential feature of our techniques is their efficiency. NEUROCARTOGRAPHY calculates the relationships of all neuron pairs in linear time, as opposed to the traditionally used quadratic time. Designed for large datasets, NEUROCARTOGRAPHY can handle data as extensive as the ImageNet dataset, which contains 1.2M images. The interface of NEUROCARTOGRAPHY tightly integrates the scalable summarization techniques, visualizing the relationships between concepts in a 2D space while also providing overviews of neuron groupings. Through extensive human evaluation, we demonstrate that NEUROCARTOGRAPHY identifies neuron groupings that correspond to clear, human-understandable concepts. And through usage scenarios, we describe how NEUROCARTOGRAPHY enables interesting and surprising discoveries, such as concept cascades of related and isolated concepts. The NEUROCARTOGRAPHY visualization runs in modern browsers and is openly available as an open-source resource.

3.1 Introduction

Deep Neural Networks (DNNs) have demonstrated remarkable success in many applications, such as object detection [81, 82], speech recognition [83, 84], and data-driven health

care [85, 86]. However, they are often considered *opaque* due to their complex structure and large number of parameters. To help practitioners and researchers more confidently and responsibly deploy machine learning models, there have been major efforts and calls from both government and industry to enable greater model interpretability [79, 78]. There is research that aims to explain models' predictions based on the inputs, such as regions of input images that contribute the most to the models' predictions [62, 47, 87, 45, 42]. However, such techniques often do not describe *how* and *where* the input features are used within the model. Recent research posits a key step towards answering these questions is to interpret *neurons* (also called *channels*), since they are highly activated for specific features from the input data [88, 56, 78, 89, 90]. While *neuron-level* interpretation may be a promising approach to discover insights, inspecting individual neurons can be time-consuming; furthermore, individual inspection does not easily reveal how *multiple* neurons may detect the same features, which means users could easily miss the bigger picture of the DNNs' decision-making processes. For example, Fig 3.2 shows that the "dog face" concept is detected by multiple neurons in InceptvionV1 model. Although it is a well-documented phenomenon that multiple neurons detect similar concepts [91] (especially in model pruning research [92, 93, 94, 95]), there is a lack of research in (1) developing scalable summarization techniques to discover concepts collectively learned by multiple neurons, and (2) enabling users to interactively interpret such concepts and their similarities. NEUROCARTOGRAPHY aims to fill this critical research gap.

Contributions. In this work, we contribute:

- **NEUROCARTOGRAPHY, an interactive system that scalably summarizes and visualizes fundamental concepts** that contribute to the behaviors of large-scale image classifier models (Fig 3.1), such as InceptvionV1 [27]. NEUROCARTOGRAPHY automatically discovers groups of neurons that detect the same concepts and describes how such neuron groups interact to form higher-level concepts and the subsequent predictions (Sec 3.6).
- **Two scalable concept summarization techniques**: (1) *neuron clustering* groups neurons based on the semantic similarity of the concepts that neurons detect (e.g., neurons detecting "dog faces" of different breeds are grouped); and (2) *neuron embedding* encodes the associations between related concepts based on how often they co-occur (e.g., neurons detecting "dog face" and "furry body" are placed closer in the embedding space, as seen in Fig 3.1A, B). Both efficient techniques avoid naively comparing all neuron pairs, resulting in a time complexity that is linear to the number of neurons, rather than quadratic time. Our techniques scale to large data, such as ImageNet ILSVRC 2012 with 1.2M images [51] (sec 3.5).
- **Interactive exploration of Concept Cascade** enables users to selectively initialize and examine how a concept detected by a neuron group would trigger higher-level concepts across subsequent layers in a neural network. NEUROCARTOGRAPHY visualizes the user-selected concept's "cascading effect," helping users interpret the successive concept

initiations and relationships (Sec 3.6.2).

- **Empirical findings through large-scale human evaluation and discovery scenarios.** Through a large-scale human evaluation, we demonstrate that NEUROCARTOGRAPHY detects neuron groups representing coherent concepts with consistent meaningful human interpretations (Sec 3.7). We describe how NEUROCARTOGRAPHY can help discover several interesting and surprising findings through usage scenarios, like identifying concept cascades for related classes (e.g., dogs of different breeds) and identifying isolated concepts that are unrelated to all other concepts in a neural network (Sec 3.8).
- **An open-sourced, web-based implementation** that helps broaden people's access to neural network interpretability research without the need for advanced computational resources. Our code and data are open-sourced[1], and the system is available at the following public demo link: https://poloclub.github.io/neuro-cartography/.

3.2 Design Challenges

Our goal is to build an interactive visual summarization of concepts learned by neural networks. Concretely, we aim to help users better understand what concepts are represented internally by groups of neurons, and how these concepts are transformed into the final prediction through interactions among neuron groups. We identify the following five design challenges (**C1** - **C4**) associated with developing our summarization techniques and designing NEUROCARTOGRAPHY.

C1 **Discovering neurons that detect similar concepts.** Existing research on DNN interpretability tends to focus on inspecting individual neurons [78, 56, 55]. While helpful, neuron-level inspection cannot easily reveal how clusters of neurons may detect the same concept, even though it is common for multiple neurons to detect similar features [92, 93, 94, 96]. As a result, users can easily miss higher-order interactions that explain how DNNs operate.

C2 **Understanding the associations between related concepts.** Interpreting individual features represented by a neural network can be useful to understand what the model sees from input data [56, 55]. However, a model doesn't base its prediction on a single feature from the input. Instead, the final prediction is often an amalgamation of multiple concepts detected by the model. This raises fundamental questions about the associations between related concepts. For example, when a concept is detected by a neural network (e.g., "dog face"), what other concepts are likely to be detected at the same time, and how are they related (e.g., would "dog tail" and "dog leg" be strongly related to "dog face")?

[1] https://github.com/poloclub/neuro-cartography

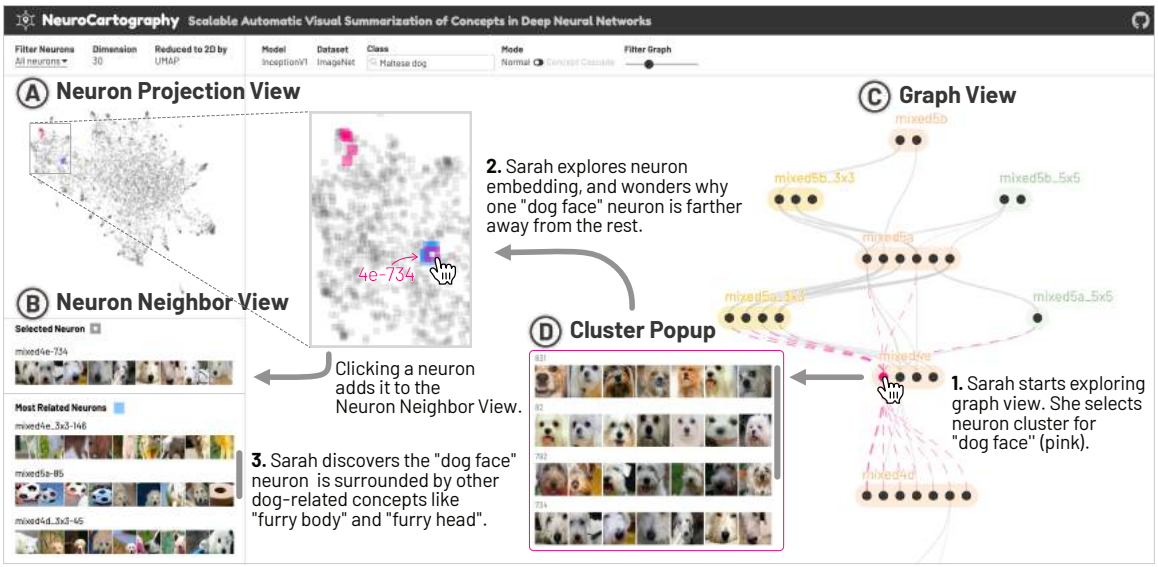


Figure 3.1: NEUROCARTOGRAPHY scalably summarizes concepts learned by deep neural networks, by automatically discovering and visualizing groups of neurons that detect the same concepts. **1.** Our user Sarah starts exploring which neuron clusters (shown as circles) play an important role for InceptionV1 to predict the *Maltese dog* class, through the *Graph View (at C)*, which clusters neurons based on the semantic similarity of the concepts detected by the neurons, and shows how those concepts interact to form higher-level concepts. Selecting the neuron cluster for "dog face" (in pink) visualizes its member neurons' features with example patches in the *Cluster Popup* (at D). **2.** Member neurons are highlighted in the global *Neuron Projection View* (at A), which summarizes all neurons' concepts from all layers by projecting them on a 2D space, placing related concepts closer together; Sarah wonders why one "dog face" neuron (#734) is farther away from the rest. **3.** Adding that neuron to the *Neuron Neighbor View* (at B) enables discovery of the most related neurons (nearby blue neurons in projection), such as "furry body" (neuron 4e_3x3-146) and "furry head" (4d_3x3-45), suggesting that the proximate projection region is in fact capturing dog-related concepts. Concepts of neurons and neuron groups are manually labeled.

C3 Scaling up concept summarization to all classes, neurons, and large datasets. Recent research in our visualization community has started to prioritize scalability to support large datasets [78, 97]. However, understanding how those approaches may extend to enable the discovery of groups of similar neurons—and encode semantic relatedness of concepts detected by the neurons—remains unclear. In the context of our work, computing neuron relationships can be computationally expensive. A naive algorithm to measure the neuron similarity would require comparing all neuron pairs, which is computationally expensive (i.e., time complexity is quadratic in the number of neurons). This naturally leads us to the question: how can we more efficiently support grouping neurons and encoding concept relatedness for complex DNNs?

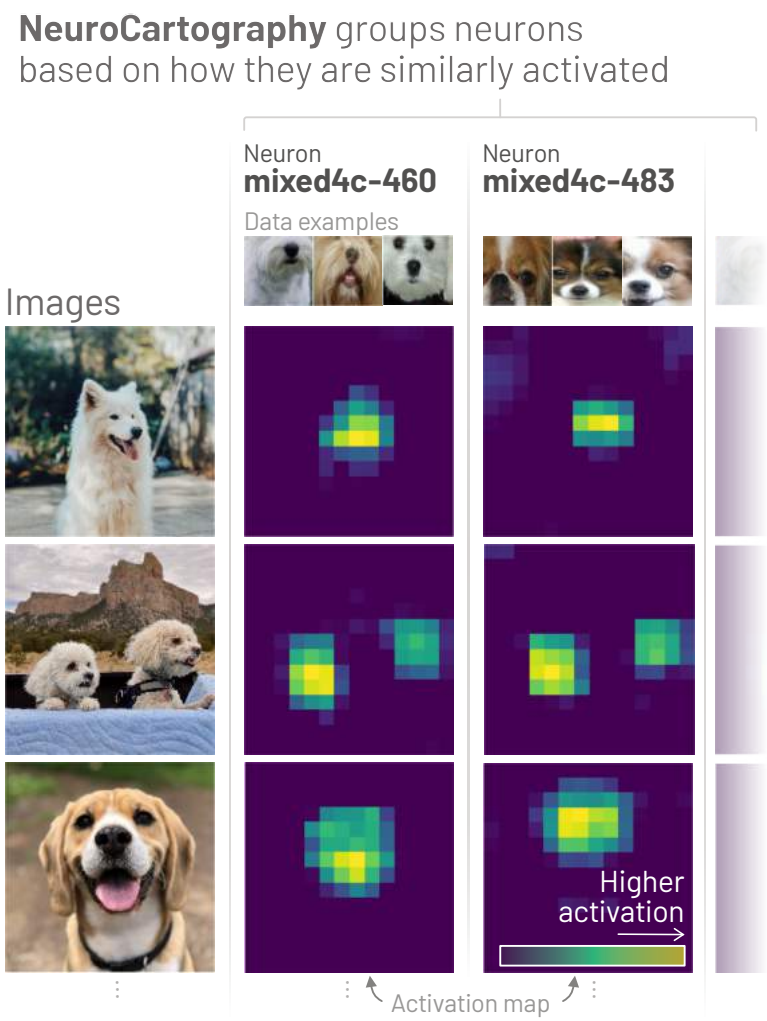


Figure 3.2: To summarize the concepts learned by a DNN, NEUROCARTOGRAPHY groups neurons based on how similarly they are activated, e.g., by the "dog face" concept. Here, neurons 460 and 483 in layer mixed4c of InceptionV1 model are similarly activated by the "dog face" concept, and are grouped in the same cluster by our approach.

C4 Understanding concept influence in a network. A promising approach to interpret a model's internal behaviour involves understanding how the model detects and combines features during inference [58, 69]. Recent research has proposed approaches to help users interpret how features may be connected [78, 89], but these approaches are performed at the neuron level, and are limited to only analyzing the relationships of neurons across two adjacent layers, instead of across the whole network. To this end, our work aims to answer a broader set of questions: How can a group of neurons detecting a concept trigger other concepts across the connections and layers of a DNN? Furthermore, how can we design an interactive visualization to support flexible exploration of such a "concept cascade"?

3.3 Design Goals

We distill the design challenges identified in Sec 3.2 to the following design goals (**G1**-**G4**) that guide NEUROCARTOGRAPHY's development.

G1 Clustering similar neurons based on activation overlap. We aim to address a major research gap in existing work by developing techniques to discover neurons that detect the same features (**C1**). Specifically, we build on prior research findings that neurons tend to selectively respond to certain input features; in the context of DNNs, this means such neurons' *activation maps* have larger values at locations where the feature is present in the input image [98, 99]. Our idea is to group neurons based on how similar their activation maps are by (1) comparing the locations of the highly-activated values in the maps (Sec 3.5.1) and (2) visualizing the concepts that are detected by such neuron groups (Sec 3.6.2).

G2 Encoding concept associations between related concepts. We aim to analyze and visualize how concepts are related based on how often they co-occur (**C2**). Our intuition is that neurons detecting highly related concepts (e.g., "dog face", "dog tail") are frequently co-activated. We aim to preserve these concept associations by learning vector representations for neurons that detect concepts associations on large image datasets (Sec 3.5.2). Furthermore, we visualize the concept embedding to enable users to interactively explore and understand related concepts across different level of abstraction such as "dog face" (higher level) and "dog eyes" (lower level) (Sec 3.6.1).

G3 Scalable summarization of concepts learned by a neural network. We aim to scale up neuron clustering (**G1**) and neuron embedding (**G2**) techniques to all neurons and all images by avoiding an explicit comparison of all neuron pairs (Sec 3.5). For scalable neuron clustering, we aim to project neurons' activation patterns into a reduced dimension and hash neurons into buckets by using these reduced projections as the key. Using this technique, we can hash similar neurons in the same bucket with high probability; more importantly, we can do this in time **linear** to the number of neurons instead of quadratic (Sec 3.5.1). For scalable neuron embeddings, we aim to subsample neuron pairs and use the sampled pairs to learn neuron vectors that will preserve the general properties of concept relatedness. From these vectors, we can infer concept relatedness of any neuron pair without comparing them directly (Sec 3.5.2).

G4 Interactive interface to explore Concept Cascade We aim to design and develop an interactive interface that enables users to selectively initialize and examine how a concept detected by a neuron group would trigger higher-level concepts across subsequent layers in a neural network **C2**. NEUROCARTOGRAPHY visualizes the user-selected concept's cascade effect, helping users interpret the successive concept initiations and relationships (Sec 3.6.2).

3.4 Model Choice and Background

In this work, we demonstrate our approach using InceptionV1 [57], a prevalent, large-scale image classifier that achieves top-5 accuracy of 89.5% on the ImageNet dataset that contains over 1.2 millions images across 1000 classes. InceptionV1 consists of multiple inception modules of parallel convolutional layers. In each module, there are four layers: an input layer, an intermediate layer where kernels' size are 3x3, another intermediate layer where kernels' size are 5x5, and an output layer. Each inception module is given a name of the form "mixed{number}{letter}," where the {number} and {letter} denote the location of a layer in the network. In InceptionV1, there are 9 such modules: mixed3{a,b}, mixed4{a,b,c,d,e}, and mixed5{a,b}. The input and output layer are given by the module name. For the intermediate layers, an suffix of either 3x3 or 5x5 is appended. For example, mixed3b is an earlier input layer and mixed3b_3x3 is an intermediate layer. While there are more technical complexities in each inception module, we follow existing interpretability literature and consider the 9 mixed layers as the primary layers of the network [56, 100]. Although this work uses specific architectural choice, the proposed summarization and visualization techniques are general and can be applied to other neural network architectures in other domains.

3.5 Scalable Neural Network Summarization

NEUROCARTOGRAPHY introduces two new scalable summarization techniques: (1) *neuron clustering* groups neurons based on the semantic similarity of the concepts detected by those neurons, and (2) *neuron embedding* encodes the associations between related concepts based on how often they co-occur. NEUROCARTOGRAPHY leverages these techniques to summarize concepts learned by neural networks. We formulate neuron clusterings in Sec 3.5.1, describe neuron embeddings in Sec 3.5.2, and detail how we filter concepts that are important for the prediction of each class in Sec 3.5.3.

3.5.1 Neuron Clustering

We aim to discover groups of neurons within the same layer that detect the same concepts. Our main idea is to group neurons that have similar activation maps. A neuron's activation map is a 2D image representing the neuron's response to an input instance, computed by the convolution of a trained kernel applied to the previous layer. A neuron's activation map reflects features detected by a neuron, showing increased values in regions of the map where detected features exist. Thus, if two neurons have similar activation maps, where highly activated areas of two activation maps largely overlap, we group the neurons. For example, in Fig 3.2, two neurons 460 and 483 in layer mixed4c of InceptionV1 are grouped

by our approach, since their activation maps have high values on similar areas. Our neuron clustering approach has two phases. First, in the preprocessing stage, we cluster neurons quickly and efficiently without looking at neurons' activation maps in detail. Next, in the main clustering stage, we further divide the preprocessed neuron groups based on the degree of overlap in the neurons' activation maps.

Preprocessing: Group Neurons Based on Common Preferred Images

The preprocessing stage aims to efficiently and quickly cluster neurons before comparing neurons' activation maps in detail. Our main idea is to group neurons if they are highly activated by many common images. For each neuron i, we first find a set of k images that maximally activate i. We sort the images by the maximum value of activation maps of i for given those images, and take the first k images. We denote the set of top k images for i as X_i. For two neurons i and j, we define their similarity as the Jaccard similarity of X_i and X_j as follows.

Definition* 1 *Concept Similarity Based on Top Images. Given two neurons i and j, and the neurons' top image sets X_i and X_j, we define the similarity of i and j as the Jaccard similarity between X_i and X_j. This value is 0 when the two image sets are disjoint, and 1 when they are equal. Neurons i and j are more similar when the Jaccard similarity is closer to 1. We formally define the similarity of i and j in Eq. (3.1)

$$SimTopImgs(i,j) = \frac{|X_i \cap X_j|}{|X_i \cup X_j|} \tag{3.1}$$

To scalably group neurons based on the common image sets, NEUROCARTOGRAPHY uses two techniques: (1) *Min-Hashing* efficiently approximates the Jaccard similarity between two neurons' top image sets; (2) *Locality-Sensitive Hashing (LSH)* efficiently hashes similar neurons in terms of the Jaccard similarity into the same buckets with high probability. It is a popular technique to use Min-Hashing and LSH to efficiently estimate Jaccard similarity between two sets and find sets of similar items, due to their scalability and theoretical guarantees on the accuracy of finding nearest neighbors [101, 102, 103, 104, 105].

Min-Hashing. It is computationally costly to measure the Jaccard similarity between large sets due to the expensive set intersection and union operations that Eq. (3.1) involves. Min-Hashing [101] efficiently estimates the Jaccard similarity. Let h be a hash function that randomly maps the entire items $\{1, ..., N\}$ to $\{1, ..., N\}$ in one-to-one correspondence. Let h_{min} be a min-hash function that outputs the minimum value retrieved from the function h: for a set S, $h_{\text{min}}(S) = \min_{s \in S}(h(s))$. The key property of Min-Hashing is that the probability of the h_{min} values of two sets being equal is equal to the Jaccard similarity between the sets. Formally, for two sets S_1 and S_2, $Pr[h_{\text{min}}(S_1) = h_{\text{min}}(S_2)]$ = Jaccard Similarity

between S_1 and S_2. For each neuron i, we define I_i as the index of images in X_i. By using the theoretical property of Min-Hashing, $Pr[h_{\min}(I_i) = h_{\min}(I_j)] = SimTopImgs(i, j)$.

Locality-Sensitive Hashing (LSH). Min-Hashing efficiently estimates the similarity of two neurons' top common images. However, it is still computationally expensive to measure the similarity of all neuron pairs. LSH is a scalable technique that finds reasonable approximations for grouping similar items without comparing all item pairs [104, 106, 107]. For each neuron i and its top images' index set I_i, we produce n Min-hash values $h_1(I_i), ..., h_n(I_i)$ with n hash functions $h_1, ..., h_n$. Then we partition the n values into b bands, each consisting of r values, such that $n = b \times r$. For each band, we hash neurons into the same buckets where r hash values of such neurons are identical. Then we finally cluster i and j in the same group, if they appear in the same bucket in at least one band. Theoretically, the probability that neuron i and j will hash to the same bucket in at least one of the b bands is $1 - (1 - s^r)^b$, where s is the true Jaccard Similarity between I_i and I_j [107].

Main Clustering: Group Neurons Based on Overlap of Activation Maps

While the preprocessing stage offers an efficient approach for preliminary neuron grouping, the main clustering stage performs finer clustering based on overlap of activation maps. In the preprocessing stage, for example, neurons for "cars" and neurons for "roads" might be grouped together, as those concepts may frequently co-occur in the same images. The main clustering stage further divides these neurons into different groups based on the concepts encoded in the activation map of the neurons. Within a preprocessed group, we finally cluster neurons in the same group, if highly activated part of the neurons' activation maps overlap significantly. We formally define the similarity of neurons i and j used in the main clustering stage as follows.

Definition 2 Concept Similarity Based on Activation Map. Given an input image $\mathbf{x}$ and two neurons i, j in the same layer, we denote their activation map as $Z_i(\mathbf{x})$ and $Z_j(\mathbf{x})$. To take only highly activated parts in each activation map, we quantize the activation maps as $Q_i(\mathbf{x}) = Z_i(\mathbf{x}) > 0$ and $Q_j(\mathbf{x}) = Z_j(\mathbf{x}) > 0$, where the quantized activation maps are a boolean matrix (i.e., true means high activation). We define the concept similarity of i and j in Eq. (3.2), where $\wedge$ and $\vee$ are element-wise `and` and `or` operation respectively, and `numTrue`$(\cdot)$ returns the number of true values in the input matrix. If `numTrue`$(Q_i(\mathbf{x}) \vee Q_j(\mathbf{x})) = 0$, the similarity between i and j is defined as 0.

$$SimActMap(i, j) = \frac{\texttt{numTrue}(Q_i(\mathbf{x}) \wedge Q_j(\mathbf{x}))}{\texttt{numTrue}(Q_i(\mathbf{x}) \vee Q_j(\mathbf{x}))} \tag{3.2}$$

The similarity $SimActMap(i, j)$ in Eq. (3.2) can be interpreted as the Jaccard similarity of highly activated parts in activation maps of i and j. We leverage Min-Hashing and

LSH in the main clustering phase again for improved scalability. For each neuron group G created in the preprocessing stage, we sample t images from the union of every belonging neuron's top k images. We denote the set of such sampled images as X_G. Formally, $X_G = \texttt{sample}(\cup_{i \in G} X_i, t)$, where $\texttt{sample}(S, t)$ randomly samples t items in set S. The main reason we use the sampled images (instead of all images) is that using all images is not very useful; because neurons selectively respond to only some images, the neurons are not activated at all by many images. To compare the similarity of two neurons, we only consider cases where both neurons are highly activated. Thus, we sample images from the union of top k images produced in the preprocessing stage, which includes many images that are likely to activate many neurons in the group G. For each group G produced in the preprocessing phase and for each image $\mathbf{x} \in I_G$, we run LSH to further group neurons based on the activation map. Then we finally group two neurons in the same bucket if the two neurons are hashed in the same bucket for least one image in I_G.

Hyperparameter Selections for Neuron Clustering

Our neuron clustering approach uses a few hyperparameters, where the total number of neurons in InceptionV1 (n) is 6,860: t is the maximum number of sampled images for each preprocessed neuron group, k is the number of top images (among 1.2M images) for each neuron, b is the number of bands in LSH, and r is the size of the bands. t helps reduce runtime through sampling; a larger value means using more samples (thus longer runtime). We experimented with values in $[50, 200]$ and observed little change in the results, thus we decided on $t = 100$. A larger k increases the chances of discovering more neuron pairs that are similarly activated. However, a value that is too large (e.g., 1M) means most neurons would have highly similar or identical sets of top images. We decided on $k = 200$, the highest value that provided good clustering while keeping total runtime reasonable. A larger b provides more opportunities to group neurons that have high Jaccard similarities. For preprocessing, we experimented with values in $[5, 2500]$ and the clustering results stabilized after b reached 1500, thus we used $b = 2000$. For main clustering, we experimented with values in $[5, 32]$, and used $b = 20$ as clustering results did not change beyond that. A larger r allows us to prune neuron pairs with low Jaccard similarities. However, a value that is too large could prune neuron pairs even if they have high Jaccard similarities. Thus, we aimed to pick a value that is not too large, or too small. We experimented with values in $[2, 5]$ for preprocessing, and $[2, 30]$ for main clustering, and found the "middle" values of 3 and 15, respectively, provided good coherence among examples image patches in the cluster results.

3.5.2 Neuron Embedding

To encode associations between concepts detected by neurons, we learn neuron embeddings that preserve the relatedness of such concepts, where neurons that detect more related concepts are located closer in the embedding space. Our embedding approach consists of two steps.

- **Step 1.** Learn vector representations of all neurons to encode relatedness among neurons' concepts. (Sec 3.5.2)
- **Step 2.** Reduce the dimensions of the learned vector representation to 2D for visualization (Sec 3.5.2), which we will describe in Sec 3.6.1.

The decision to adopt a two-step approach to first generate higher-dimensional vector representations for neurons (Step 1) was motivated by prior work [108, 109, 110], where abstract concepts are better captured by higher-dimensional representations, which opens up the possibilities for supporting interpretation tasks at higher fidelity.

Step 1: Encode Relatedness of Neurons' Concepts via Vector Representations

The objective function J to minimize of our embedding approach is defined in Eq. (3.3), where D is a set of sampled neuron pairs detecting highly related concepts, V_i is the embedding vector of neuron i to learn, and $\sigma(x)$ is the sigmoid function (i.e., $\sigma(x) = 1/(1+e^{-x})$).

$$J = \sum_{(i,j)\in D} -\log(\sigma(V_i \cdot V_j)) \tag{3.3}$$

The objective function induces the embedding vectors V_i and V_j of neurons i and j detecting highly related concepts to yield high $\sigma(V_i \cdot V_j)$. A large value of the dot product of two vectors indicates that the vectors are far from the origin in the same direction. The sigmoid function controls the magnitude of dot product, so that those vectors do not move too far away from the origin. Thus, the objective function induces vectors of highly related neurons to be located closely and moderately far away from the origin. Our use of cross-entropy loss was motivated by prior work [110] where no predefined classes or labels are available, which is the case here (i.e., no concept labels for each neuron).

To sample neurons of highly related concepts, we find neurons that are frequently co-activated. We reuse the top k images for each neuron which are obtained at the preprocessing stage of neuron clustering (Sec 3.5.1). For each image, we first generate a list of neurons that have such image in the neuron's top k images. Then, we sample neuron pairs from the top neuron list. We first randomly shuffle the neuron list, apply a sliding window of size 2 on the shuffled list, and sample pairs of neurons that are co-occurred in the sliding window. A good property of using sampled neuron pairs instead of all pairs is that sampled pairs indirectly can imply the relation of all neuron pairs. If two pairs (a, b) and (b, c) of

highly related items are sampled, we can infer that (a, c) is also highly related. Our embedding approach efficiently learns and preserves such indirect concept relatedness, as well as direct relatedness straightly from the sampled pairs. Note that the number of sampled pairs is linear to the number of neurons, not quadratic. This sampling approach results in the training data of size linear to the number of neurons, and the time complexity of our neuron embedding method is linear to the number of neurons.

To further speed up the optimization process, we use negative sampling approach: concretely, we find pairs of non-related neurons and induce their embedding vectors far apart. For a given neuron, we find another non-related neuron by randomly sampling one among all neurons in the same layer. The new objective is defined in Eq. (3.4), where M is the size of negative sampling for a pair of related neurons (i, j).

$$\begin{aligned} J = \sum_{(i,j)\in D} \Bigg(& -\log(\sigma(V_i \cdot V_j)) \\ & + \sum_{m=1}^{M} \big(-\log(1-\sigma(V_i \cdot V_m)) - \log(1-\sigma(V_j \cdot V_m)) \big) \Bigg) \end{aligned} \tag{3.4}$$

We use gradient descent to learn neuron vector representations that optimize J. The derivatives of objective function J with respect to the neuron vector V_i and V_j are as in Eq. (3.5) and (3.6).

$$\frac{\partial J}{\partial V_i} = -\big(1-\sigma(V_i \cdot V_j)\big) V_j + \sum_{m=1}^{M} \sigma(V_i \cdot V_m) V_m \tag{3.5}$$

$$\frac{\partial J}{\partial V_j} = -\big(1-\sigma(V_i \cdot V_j)\big) V_i + \sum_{m=1}^{M} \sigma(V_j \cdot V_m) V_m \tag{3.6}$$

We update the embedding by gradient descent as in Eq. (3.7), where γ is the learning rate. Further algorithm details are described in Appendix A.

$$V_i \leftarrow V_i - \gamma \frac{\partial J}{\partial V_i}, \qquad V_j \leftarrow V_j - \gamma \frac{\partial J}{\partial V_j} \tag{3.7}$$

Step 2: Dimensionality Reduction

To project neurons' vector representations learned in the previous step onto a 2D space, we use UMAP, a non-linear dimensionality reduction technique that preserves global data structures and local neighbor relations [111].

Hyperparameter Selection for Neuron Embedding

Our neuron embedding uses a few hyperparameters: size of negative sampling M, number of epochs, and learning rate γ for training. Tuning M helps prevent overfitting; a value that is too large could introduce significant noise during training. Epochs and learning rate affect the training runtime and quality. More epochs usually causes more distinct clusters to form. We experimented with different combinations of hyperparameter values and found these chosen ones provide a good visual results: M=10, epoch=30, γ=0.01.

3.5.3 Filtering Each Class's Important Model Substructures

Important Neurons and Neuron Groups

Our goal is to summarize important model substructures (i.e., neuron groups) that contribute to a model's class prediction. To do so, we follow an approach similar to [78], and adapt our implementation to include neuron groups. The importance of each neuron i in layer l for the prediction for a class c is computed as the number of images of c by which i is maximally activated. Whether a neuron i in layer l is highly activated for an image $\mathbf{x}$ is decided by the maximum value of activation map of i for $\mathbf{x}$ (i.e., $\max(Z_i^l(\mathbf{x}))$). For an image $\mathbf{x}$ for a class c, we find 5 most activated neurons for each layer as suggested in [78].

After obtaining the importance of each neuron for a class c, we then compute importance of each neuron group for c. For a neuron group G, we take 10 neurons at most that have the highest importance for c, to avoid overcrowding the visualization, while we also observe that 10 neurons are enough to explain what concepts the group is detecting. We then compute the importance of G for c as the average of importance score of at most the top 10 neurons.

Important Connections among Neurons and Neuron Groups

Important neurons and neuron groups summarize concepts that are important for a class prediction. We further want to describe how those concepts interact to form higher level abstractions, by representing the connections between the model substructures. We follow similar steps in [78] to compute the influence from each neuron in a given layer to each other neuron in a successive layer. At a high-level, for a given class c, the influence of each neuron-neuron connection is the number of images of c that use such connection as a major path to transmit high activation signal. Thus, if two neurons have strong influence values, it means that the neuron in an earlier layer activated the other neuron in a subsequent layer for many images of the selected class. Finally, to detect if an image uses a connection as a major path, we compute the maximum convolution value of activation map corresponding to the source neuron multiplied with a slice of learned kernel tensor between the two

neurons. We refer the reader to [78] for a more in-depth treatment of this approach. Our implementation deviates from [78] in the last step, when aggregating influence values for each neuron group. For two neuron groups $G1$ and $G2$, we compute average of the influence values between any neuron in $G1$ and any neuron in $G2$, and use such average value as the connection weight between $G1$ and $G2$.

3.6 User Interface

Based on our design goals (Sec 3.3) and our neural network summarization techniques (Sec 3.5), we present NEUROCARTOGRAPHY, an interactive system that summarizes concepts detected by neuron clusters (Fig 3.1). The NEUROCARTOGRAPHY interface consists of three components: (1) the header shows metadata and contains a few controls for the Graph View, (2) Neuron Projection View and Neuron Neighbor View provides a global overview of all neurons (Sec 3.6.1), and (3) Graph View visualizes concept relations (Sec 3.6.2).

3.6.1 Neuron Projection View and Neuron Neighbor View: Global Overview of Neurons' Concept

The Neuron Projection View (Fig 3.1A) aims to show a global overview of all neurons in a model, such that neurons detecting more related concepts are positioned closer. We project embedding of all neurons computed in Sec 3.5.2 onto a 2D space.

In the Neuron Projection View, each neuron is represented as a rectangle. Hovering over a neuron shows representative example patches to explain such neuron's concepts (Fig 3.3). Users can focus on a neuron by clicking the corresponding rectangle. As a result, the selected neuron is marked with inner white rectangle, and the selected neuron's neighbors are highlighted with blue in the embedding space. Also, at the bottom left, Neuron Neighbor View displays the neighbor neurons with their example patches (Fig 3.3). At the top, Neuron Projection View provides multiple filtering options, such as showing all neurons, neurons for a selected class, and neurons for selected clusters. Users can freely zoom and pan in the view.

3.6.2 Graph View: Neuron Clusters and their Interaction

The goal of Graph View is to visualize what concepts are detected by which clusters of neurons, and how those clusters collaborate to form higher-level concepts and the final prediction (Fig 3.1C). Graph View provides two modes through toggle button in the header: (1) class exploration mode (Sec 3.6.2) to visualize concepts important for a user-selected class and (2) concept cascade mode (Sec 3.6.2) to selectively activate a concept and examine its cascade effect.

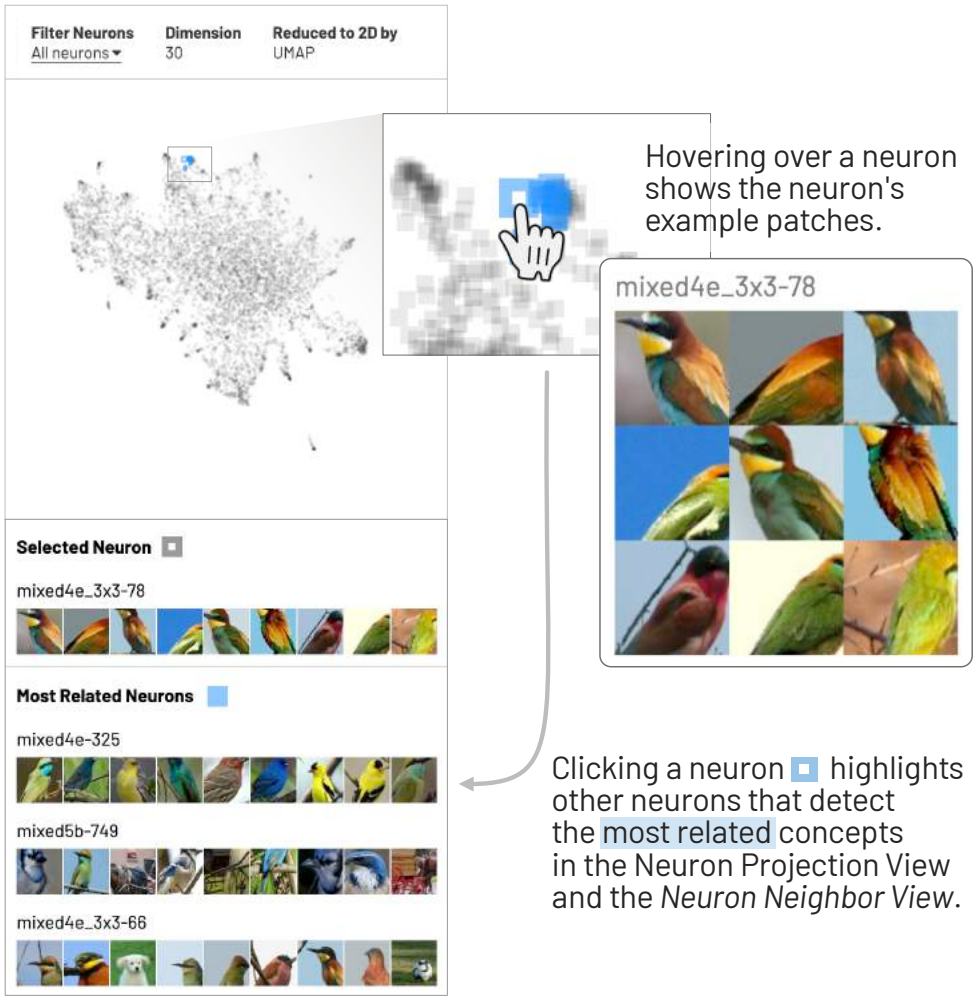


Figure 3.3: Neuron Projection View visualizes associations between related concepts based on co-occurrence, by projecting neurons on a 2D space where each rectangle is a neuron. Hovering over a neuron shows its example data patches in a popup. Clicking a neuron selects it, marking it with a white dot at its center. Related neurons are in blue, and related example patches are displayed in Neuron Neighbor View (at bottom left).

Class exploration

When a user selects a class in the header, Graph View shows a subgraph of the entire neural network that is relevant for the class prediction (Sec 3.5.3). The nodes (circles) are neuron clusters or individual neurons within the same layers, displayed in the order of their important scores computed in Sec 3.5.3. The edges represent influence among the nodes, where edge weights are computed in Sec 3.5.3. Thicker edges indicate more important connections. Users can filter the graph based on the importance score, using the a slider in the header. The graph visualization is shown in a zoomable and panable canvas.

When a user hovers over a node, NEUROCARTOGRAPHY first highlights the node and

the edges connected to the hovered node with pink, then displays the Cluster Popup (Fig 3.1D) view, which contains example patches. User can pin interesting nodes by clicking them. The pinned nodes are highlighted in the Neuron Projection View and the Graph View with pink. Neuron Projection View and Graph View are tightly integrated: hovering over a neuron in Neuron Projection View highlights its belonging cluster in the Graph View, and hovering over a node in Graph View highlights its member neurons in the Neuron Projection View. Users can filter neurons in the Neuron Projection View to focus on pinned nodes using the dropdown menu in the header.

Concept Cascade: Successive Concept Detection Initiated by a User-Selected Concept

Besides displaying how detected concepts interact within two adjacent layers for given images of a class, Graph View visualizes how one concept can influence multiple other concepts across all layers and classes through a concept cascade. Users can enter the concept cascade mode by toggling the button in the header (Fig 3.4, at 1). Then, users can click a concept cluster node to select it and manually activate the selected concept cluster (Fig 3.4, at 2), without feeding any input images. Clicking causes the neuron cluster to induce a concept cascade that triggers higher-level concepts across subsequent layers in the model. Manual concept stimulation involves first setting every value in the activation map of the selected cluster's member neurons to 1, then forward-feeding such activation to the next layers through existing connections between neurons. In the concept cascade, neuron clusters that are highly related to *Maltese dog*, and strongly contribute to the prediction "furry face," are included as part of the class's graph summary (Fig 3.4, left). The cascade also includes concepts that related to the Maltese dog class, but are not as important for its prediction, such as "bear face" and "black dog face." We highlight such concepts and their connections on the right side of the class's graph summary (Fig 3.4, right).

3.6.3 System Implementation

To broaden access to our work, NEUROCARTOGRAPHY is web-based and can be accessed from any modern web-browser. NEUROCARTOGRAPHY uses the standard HTML/CSS/JavaScript stack, and D3.js for rendering SVGs. We ran all our deep learning code on a NVIDIA DGX 1, a workstation with 8 GPUs (each with 32GB of RAM), 80 CPU cores, and 504GB of RAM. With this machine we could generate everything required for all 1000 ImageNet classes under 24 hours. The most computation intensive part was computing all neurons' activation maps for all images (once for determining top-k images for each neuron; once for main clustering stage; each run was about 5 hours). Each of the other processes, thanks to the scalability of Min-Hashing, LSH, and the sampling-based neuron embedding, took less than an hour.

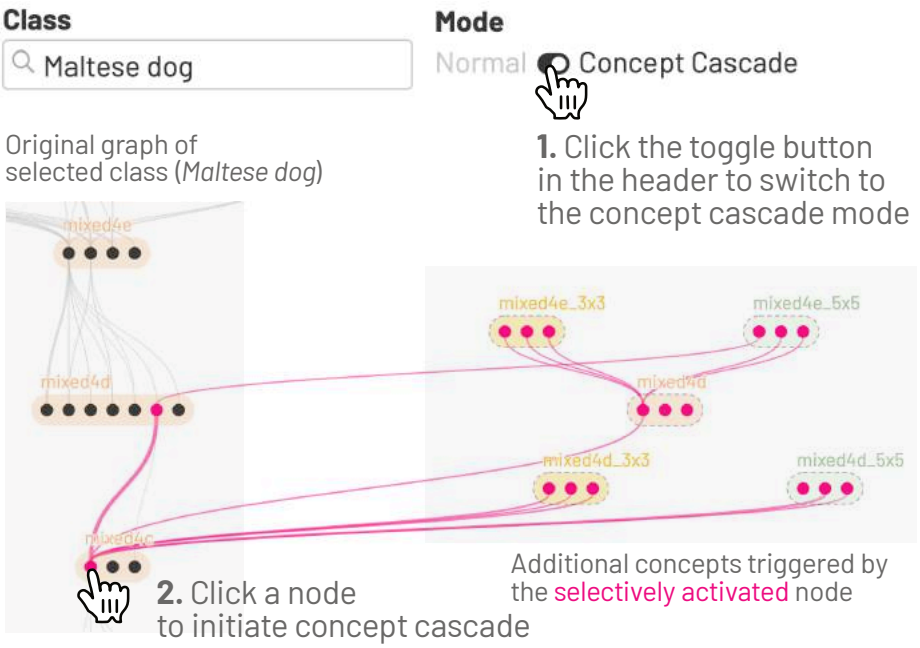


Figure 3.4: Clicking a concept cluster node activates the concept, causing the neuron cluster to induce a **concept cascade** that triggers higher-level concepts across subsequent layers in the model. **Left:** in the concept cascade, some neuron clusters are strongly contributing to current class's prediction and are shown in the class's graph summary. **Right:** Concepts less related are shown on the right hand side.

3.7 Human Experiment to Evaluate NEUROCARTOGRAPHY

To validate the human interpretability of the clusters discovered with NEUROCARTOGRAPHY, we conducted a large-scale human evaluation using Amazon Mechanical Turk, a standard practice for computer vision tasks [112], basing our experimental design on similar work in image [62] and language [113] based cluster interpretability studies. For the experiment each user was presented a series of tasks like the task shown in Fig 3.6. In each task we display the example patches for 6 neurons which are composed of 6 randomly sampled neurons, or 5 neurons from a cluster (determined either by NEUROCARTOGRAPHY or hand selected) as well as one 'intruder' neuron which is randomly sampled from the rest of the network. Users were told that each task contains either all random patches, or a cluster of 3-5 related neurons, and were asked to identify which, if any, of the shown patches form a cohesive cluster and could select any number of the given options. By not disclosing the number of intruders users are forced to only select clusters which are fully coherent among themselves (as opposed to simply finding the least representative example). We can then measure 'false positives' (where users mistakenly identify the intruder as part of the cluster) and 'false negatives' (where users may decide to not include patches from the cluster). This style of study measures the performance of human annotators against model output

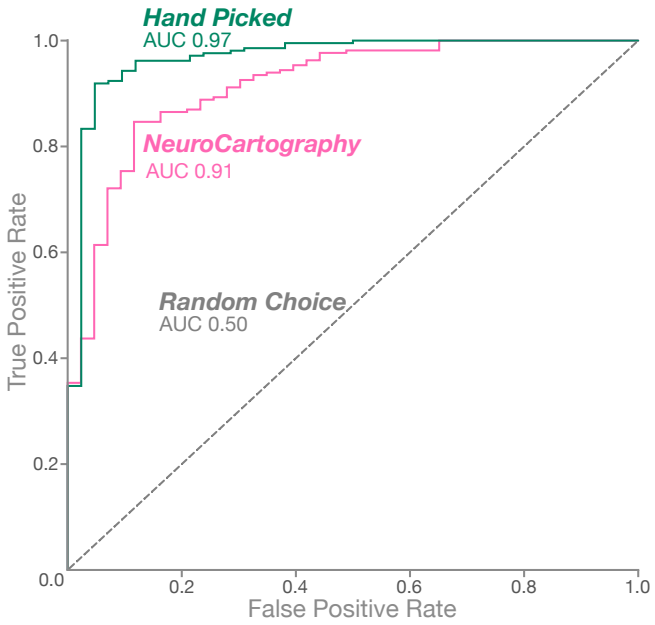


Figure 3.5: ROC Curve for user estimations of cluster inclusion. Both hand picked and NEUROCARTOGRAPHY generated clusters perform well overall, implying that the clusters generated are interpretable enough to be consistently recognised by different users.

as ground truth, the inverse of standard machine learning metrics, in order to understand the difficulty of the interpretation task that the interpretability method (in our case clusters from NEUROCARTOGRAPHY) provides. We assume that higher performance by humans equates to an easier task for humans to interpret, highlighting a better methodology for providing explainability. Our evaluation specifically takes the inverse framing of other studies which ask users to positively identify the intruder rather than the cluster [113, 62]. We do this in order to independently measure the 'false negative' class of errors that are not possible to detect using pure intruder detection.

For our study, we generated 99 unique sets of neurons such as those in Fig 3.6, of which 43% were generated using NEUROCARTOGRAPHY, 43% were generated from hand picked clusters, and 14% were generated completely at random with no underlying cluster. These sets were used to populate 9 different questionnaires of 11 sets which Amazon Mechanical Turk workers located within the U.S. completed, receiving compensation of $1 per questionnaire and taking an average time of 7 minutes to complete. An average of 42 unique workers completed each questionnaire, for a total of 3374 unique human judgements of clusters from 244 unique workers overall.

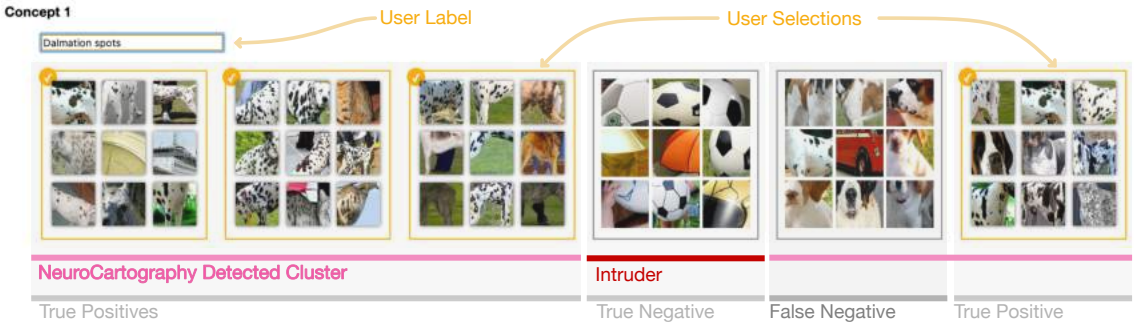


Figure 3.6: Example question from MTurk evaluation. Users were presented with six neuron patches and asked to determine if there is a coherent cluster and if so provide a short label. In this example of a NEUROCARTOGRAPHY generated cluster we can see which neuron is the out of cluster intruder, which neurons are in the cluster, which options the user selected, and the classification results of true positives for the neurons the user correctly selected, true negative for not selecting the intruder, and a false negative error for not selecting a neuron that is in the cluster.

3.7.1 Cluster Cohesion

The measure of cluster cohesion used in other intruder detection experiments is accuracy on the binary prediction task on whether a user correctly identified the intruder. In our study this is equivalent to the false positive rate for cluster inclusion predictions. For the random baseline, the false positive rate was found to be $30.6\% \pm 3.1\%$ (showing how easy it is for humans to find spurious patterns where none exist) while hand picked clusters had a false positive rate of $6.1\% \pm 1.2\%$, and NEUROCARTOGRAPHY generated clusters had a false positive rate of $11.8\% \pm 1.7\%$. With 95% confidence intervals, both clustering techniques significantly outperformed the baseline on the binary task.

However, in real applications, and in the context of this study, users have varying thresholds for how similar neurons must be in order to be included within the same cluster, even if they are using the same underlying similarity heuristic for their judgements. Because our study design gave participants a choice for the number of options to select within a given set, each neuron's inclusion is essentially an independent judgement of whether it fits within the cluster (if the user determines a cluster exists). By getting enough of these independent measurements for each neuron we can use the proportion of users who choose to include a neuron within a cluster as a score of how likely users expect it to be included, or how cohesive the specific neuron is with the the context of the whole cluster. By using this score instead of binary accuracy, we can evaluate the full space of trade-offs between false negative and false positive errors using the receiver operative characteristic (ROC) which provides a fuller, threshold-independent picture of performance, offering a more robust variation of the specific experimental setup and balance of options and intruders [114].

The score used to define the ROC in Fig 3.5 was calculated using the percentage of users including a given neuron within a cluster in the case that the user determines there is a cluster present in the set (that is they select 3, 4, or 5 neurons as opposed to 0). We used this method for both hand selected and NEUROCARTOGRAPHY generated clusters (and excluded the random baseline as it contains no true positive values). Taking the area under the curve (AUC) of the ROC for each clustering method (Fig 3.5) we find again that hand selected clusters again outperform NEUROCARTOGRAPHY, but that both perform substantially better than chance with AUC values of 0.97 ± 0.04 for hand created clusters and 0.91 ± 0.04 for NEUROCARTOGRAPHY. This result shows that NEUROCARTOGRAPHY produces clusters that are nearly as interpretable as hand crafted clusters across different inclusion thresholds, and are both much more reliably detected than chance.

3.7.2 Label Cohesion

To further understand the consistency of the patterns agreed upon by users, we asked users to describe individual clusters they selected. Without a ground truth for cluster descriptions, we looked to statistically compare how different users labeled the same clusters in order to see the consistency of the discovered concept. To compare cluster level descriptions, we rely on sentence level embeddings from the Universal Sentence Encoder (USE)[115]. USE works by projecting sentences into embeddings which can be compared (using cosine similarity) to identify the presence of similar ideas. USE similarity is preferred to word choice overlap metrics used in [62], since it captures semantic similarity of the actual concepts being discussed regardless of phrasing (for example matching labels describing the same set with "dots," "circles," and "round objects").

First, in order to build a baseline for semantic similarity values, we calculated the average pairwise similarity between all labels across different clusters throughout the dataset to be $0.301 \pm .003$. Then for each unique set, we found the average pairwise similarity between the labels from all of the users with labels for that specific set. These average within-group similarities for each class of cluster are 0.59 ± 0.05 for hand picked clusters and 0.51 ± 0.05 for NEUROCARTOGRAPHY generated clusters. Both of these results are significantly higher than the baseline, showing semantic consistency between how users understand the clusters that they detect. Complementary future evaluation may assess the degree to which the neuron groups are capturing redundant semantics (e.g., track accuracy changes as neurons are pruned).

3.8 Usage Scenarios

3.8.1 Automatically Discovering Backbone Concept Pathways for Related Classes

DNNs are known to learn general concepts: this generalizability is widely leveraged in transfer learning, model compression, and model robustness research [116, 117]. However, it is challenging to *automatically* discover and interpret which key concepts are progressively combined or connected internally in a model, or how such "backbone" concept pathways may be shared across related classes. Recent research has proposed approaches to help users interpret how features may be connected [78], but such approaches are performed at the neuron level, limited to only analyzing the relationships of neurons across two adjacent layers, instead of across the whole network. The biggest drawback is the dependence on manual processes.

NEUROCARTOGRAPHY's Concept Cascade can help automatically discover backbone concept pathways for related classes across all layers. Users can selectively activate a concept and examine the concept's cascade effect to interpret the successive concept initiations in layer layers, while identifying concepts' general relationships to related classes. For example, while inspecting the *Maltese dog* class, we found a cluster detecting "dog face" in mixed4c (Fig 3.7). Through Concept Cascade mode, we manually activate this concept to trigger and discover its related concepts in subsequent layers not only for the Maltese dog class, but also for other breeds of dogs such as *Beagle* and *Appenzeller*. Through these concept cascades, we visualize how concepts may evolve over the network, such as from the generic "dog face" concept to the specific "furry dog face" concept in later layers.

Backbone concept pathways can also be used to highlight learned generalize properties, like "curve detectors" (Fig 3.8). Existing work [118] has observed that several neurons in the earlier layers detect curves of different orientations. Even though these detectors were discovered manually, their analysis yields interesting properties of the curve concepts, like how sensitive the curve detectors are to curvature and what orientations do they respond to. Using NEUROCARTOGRAPHY, we automatically discovered more curve detectors in InceptionV1. We selected a node in mixed3b_5x5 layer which detects a curve of a specific orientation, selectively activated in the Concept Cascade mode, and discovered the curve detectors across layers such as neurons shown in Fig 3.8. We also observed that those curve detectors are clustered in the preprocessing stage of our clustering algorithm, but not grouped by the main clustering stage. This is because the curve detectors are highly selective for orientations, causing highly activated regions of the activation maps are different by the detectors.

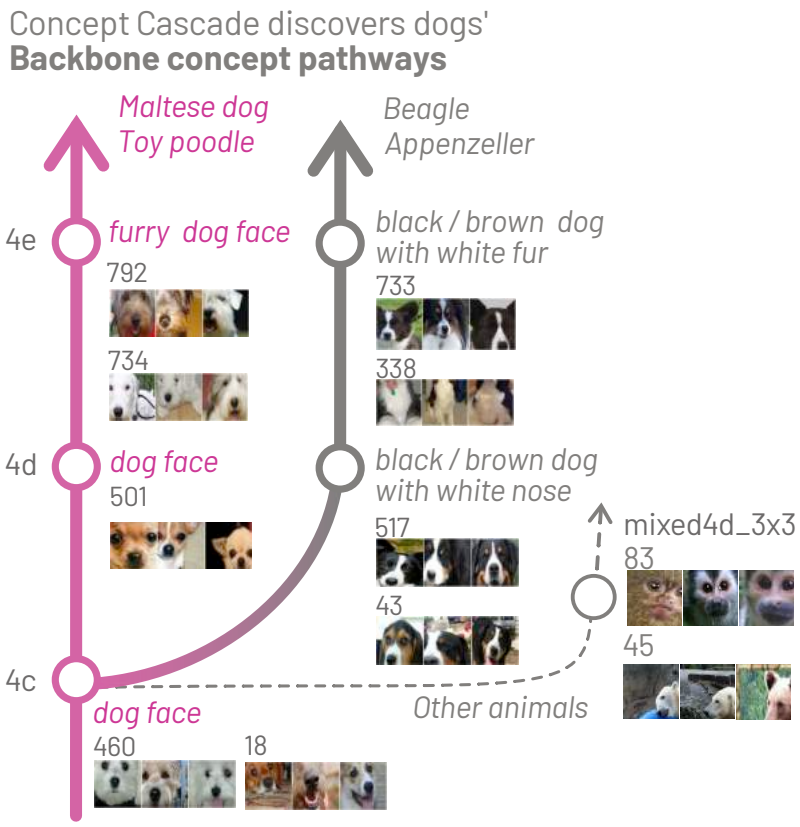


Figure 3.7: Through Concept Cascade, users manually activate the "dog face" concept to discover its related concepts in subsequent layers, not only for the current "Maltese dog" class but also for other breeds of dogs. Concept Cascade helps users visualize how concepts may evolve over the network, such as from the more generic "dog face" concept to the more specific "furry dog face" concept in later layers. Concepts are manually labeled.

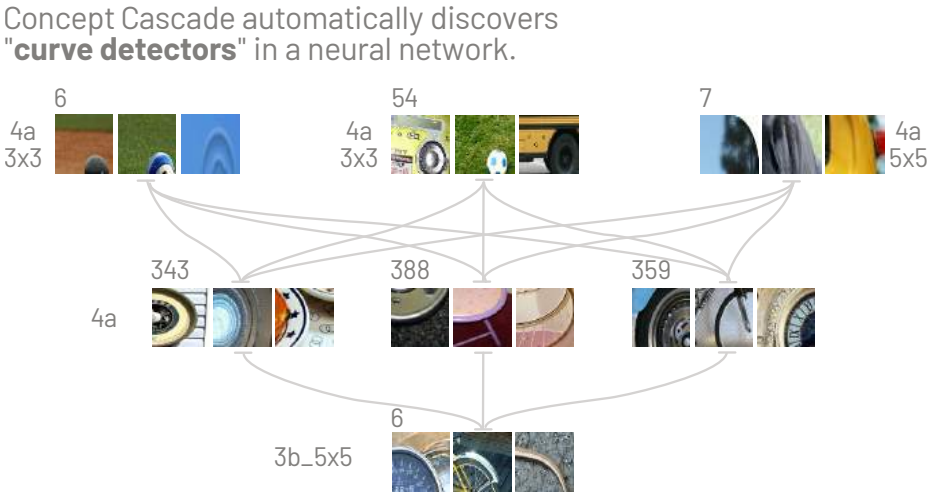


Figure 3.8: Concept Cascade automatically discovers neurons that detects curve of specific orientations, which have been manually found in [118].

3.8.2 Finding Isolated Concepts

While inspecting the Neuron Projection View, we noticed a small number of neurons are distinctively positioned very far away from all other neurons in the embedding space (Fig 3.9, right). Inspecting such isolated neurons reveals that they are detecting the "watermark" concept (see example from mixed5b-337 layer at Fig 3.9, top-left). Interestingly, as watermarks can appear on almost any kinds of images independent of the image content that the watermarks are placed above (e.g., copyright watermark can appear on an image of a car, a dog, or a pineapple), this means the neurons responsible for detecting watermarks would frequently co-activate with each other, but such "watermark neurons" co-activate relatively less so with the neurons detecting the concepts that describe the image content since watermarks are not associated with only some specific features. NEUROCARTOGRAPHY's neuron embedding algorithm is able to discover this interesting phenomenon about the watermark neurons, placing them close together to reflect the concept coherence for watermark, and away from other neurons to reflect the watermark's non-specificity for image content. NEUROCARTOGRAPHY allows us to easily verify our observations and conclusions. For example, selecting mixed5b-337, a watermark neuron (Fig 3.9, top-left), in the Neuron Projection View brings in its most related neurons in the Neuron Neighbor View (e.g.,mixed5b-86, mixed4c-342, mixed4e-296), which are all watermark neurons as well. These neurons are also clustered in the Graph View (e.g., in mixed5b layer, neurons #337, #113, #289, and #86 appear in the same neuron cluster).

3.9 Conclusion, Limitations and Future Work

We have presented NEUROCARTOGRAPHY, an interactive system that scalably summarizes and visualizes concepts learned by DNNs via scalable concept summarization techniques for *neuron clustering* and *neuron embedding*. Through a large-scale human evaluation, we have demonstrated that our techniques discover neuron groups that represent coherent, human-meaningful concepts. Our system runs in modern browsers and is open-sourced. Below, we discuss limitations of our approach and future research directions for extending this investigation.

Further dissecting poly-semantic neurons. We believe our work has taken a major step in addressing the research challenging of automatically and scalably grouping neurons that detect the same concept, going beyond manual, neuron-level inspection (e.g., [88, 78, 61, 54]) to provide a higher-level perspective for the knowledge learned by a network. Our work, however, is not designed for "dissecting" neurons that may become activated for multiple seemingly unrelated concepts, which has been observed in recent work, e.g, [58]. For example, in InceptionV1, at least poly-semantic neuron that responds to cat faces, fronts of cars, and cat legs [58]. NEUROCARTOGRAPHY cannot "split" this neuron into

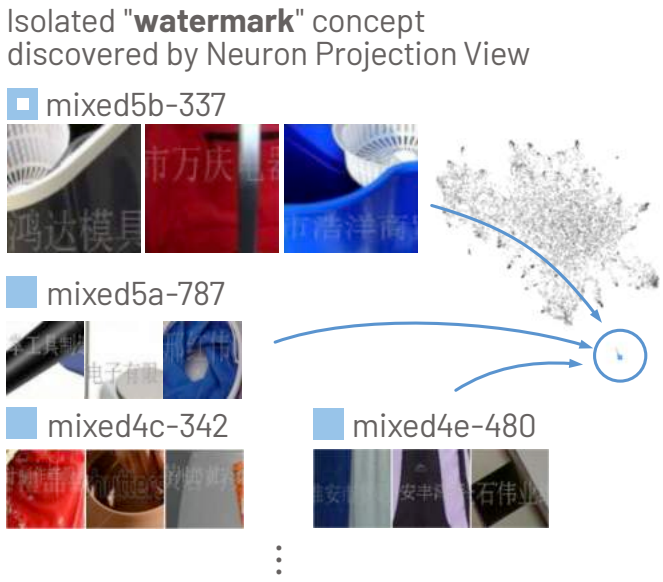


Figure 3.9: NEUROCARTOGRAPHY reveals the interesting phenomenon about the isolated "watermark" concept (example image at top-left), that watermarks are not specific to any image features (i.e., can appear on almost any kinds of images), thus watermark neurons are placed far away from all other neurons due to relatively low co-activations (see Neuron Projection View on the right).

multiple neuron, each detecting one concept and put that newly created neuron into its logical neuron cluster with other similar neurons in the network. Tackling poly-semantic neurons is an exciting and challenging direction for future work.

Integrating NEUROCARTOGRAPHY into more applications. Currently, our work focuses on using NEUROCARTOGRAPHY to enhance interpretability of DNNs. As DNNs are increasingly used in an ever-increasing variety of applications, our approaches can help practitioners and researchers assess the effectiveness of their ideas. For example, in the neural network compression community, several methods [96, 92, 93, 94] leverage potential neuron redundancies to generate compressed models while maintaining prediction accuracy. NEUROCARTOGRAPHY can help researchers interpret the semantic similarity between the compressed model and the original, uncompressed models, which helps them assess if their techniques are indeed preserving the "gist" of the knowledge important for prediction, or if they are leveraging some other features of the data of the model. Currently, concepts need to be manually labeled; automatic labeling will increase the tool's usability. Also, current Neuron Projection View presents all neurons in the same plot even though some concepts' abstraction levels could be very different; our future work includes providing users with the ability to select layers that they want to investigate. We look forward to seeing the impact that NEUROCARTOGRAPHY may contribute, from assisting evaluation of existing techniques (e.g., model compression, adversarial attacks and defenses), to developing new ones.

Visualizing other neural network models. We have justified our model choice in section 3.4; we are working to extend support to other CNN models. Our approach can easily be adapted to simpler models (e.g., VGG [33]). For more complex networks (e.g., ResNets [119], small extensions would be needed to handle more types of connections present in the network (e.g., skip connections could be represented as skip-layer edges in the graph view).

CHAPTER 4

SUMMIT: SCALING DEEP LEARNING INTERPRETABILITY BY VISUALIZING ACTIVATION AND ATTRIBUTION SUMMARIZATIONS

Deep learning is increasingly used in decision-making tasks. However, understanding how neural networks produce final predictions remains a fundamental challenge. Existing work on interpreting neural network predictions for images often focuses on explaining predictions for single images or neurons. As predictions are often computed based off of millions of weights that are optimized over millions of images, such explanations can easily miss a bigger picture. We present SUMMIT, the first interactive system that scalably and systematically summarizes and visualizes what features a deep learning model has learned and how those features interact to make predictions. SUMMIT introduces two new scalable summarization techniques: (1) *activation aggregation* discovers important neurons, and (2) *neuron-influence aggregation* identifies relationships among such neurons. SUMMIT combines these techniques to create the novel *attribution graph* that reveals and summarizes crucial neuron associations and substructures that contribute to a model's outcomes. SUMMIT scales to large data, such as the ImageNet dataset with 1.2M images, and leverages neural network feature visualization and dataset examples to help users distill large, complex neural network models into compact, interactive visualizations. We present neural network exploration scenarios where SUMMIT helps us discover multiple surprising insights into a state-of-the-art image classifier's learned representations and informs future neural network architecture design. The SUMMIT visualization runs in modern web browsers and is open-sourced.

Deep learning is increasingly used in decision-making tasks, due to its high performance on previously-thought hard problems and a low barrier to entry for building, training, and deploying neural networks. Inducing a model to discover important features from a dataset is a powerful paradigm, yet this introduces a challenging *interpretability* problem — it is hard for people to understand what a model has learned. This is exacerbated in situations where a model could have impact on a person's safety, financial, or legal status [120]. Definitions of interpretability center around *human understanding*, but they vary in the aspect of the model to be understood: its internals [121], operations [122], mapping of data [123], or representation [26]. Although recent work has begun to operationalize interpretability [124], a formal, agreed-upon definition remains open [125, 126].

Existing work on interpreting neural network predictions for images often focuses on explaining predictions for single images or neurons [45, 43, 56, 100]. As large-scale model predictions are often computed from millions of weights optimized over millions of images, such explanations can easily miss a bigger picture. Knowing how entire classes are represented inside of a model is important for trusting a model's predictions and deciphering what a model has learned [26], since these representations are used in diverse tasks like detecting breast cancer [127, 128], predicting poverty from satellite imagery [129], defending against adversarial attacks [130], transfer learning [131, 132], and image style transfer [133]. For example, high-performance models can learn unexpected features and associations that may puzzle model developers. Conversely, when models perform poorly, developers need to understand their causes to fix them [134, 26]. As demonstrated in Fig 4.1, InceptionV1, a prevalent, large-scale image classifier, accurately classifies images of ***tench*** (yellow-brown fish). However, our system, SUMMIT, reveals surprising associations in the network that contribute to its final outcome: ***tench*** is dependent on an intermediate person-related "hands holding fish" feature (right callout) influenced by lower-level features like *"scales," "person,"* and *"fish"*. There is a lack of research in developing scalable summarization and interactive interpretation tools that simultaneously reveal important neurons and their relationships. SUMMIT aims to fill this critical research gap.

Contributions. In this work, we contribute:

- **SUMMIT, an interactive system for scalable summarization and interpretation** for exploring entire learned classes in prevalent, large-scale image classifier models, such as InceptionV1 [27]. SUMMIT leverages neural network feature visualization [56, 135, 57, 42, 53] and dataset examples to distill large, complex neural network models into compact, interactive graph visualizations (Sec 4.5).
- **Two new scalable summarization techniques** for deep learning interpretability: (1) *activation aggregation* discovers important neurons (Sec 4.4.1), and (2) *neuron-influence aggregation* identifies relationships among such neurons (Sec 4.4.2). These techniques scale to large data, e.g., ImageNet ILSVRC 2012 with 1.2M images [51].
- **Attribution graph, a novel way to summarize and visualize entire classes**, by combining our two scalable summarization techniques to reveal crucial neuron associations and substructures that contribute to a model's outcomes, simultaneously highlighting *what* features a model detects, and *how* they are related (Fig 4.2). By using a graph representation, we can leverage the abundant research in graph algorithms to extract attribution graphs from a network that show neuron relationships and substructures within the entire neural network that contribute to a model's outcomes (Sec 4.4.3).
- **An open-source, web-based implementation** that broadens people's access to interpretability research without the need for advanced computational resources. Our work joins a growing body of open-access research that aims to use interactive visualization to explain complex inner workings of modern machine learning techniques [136,

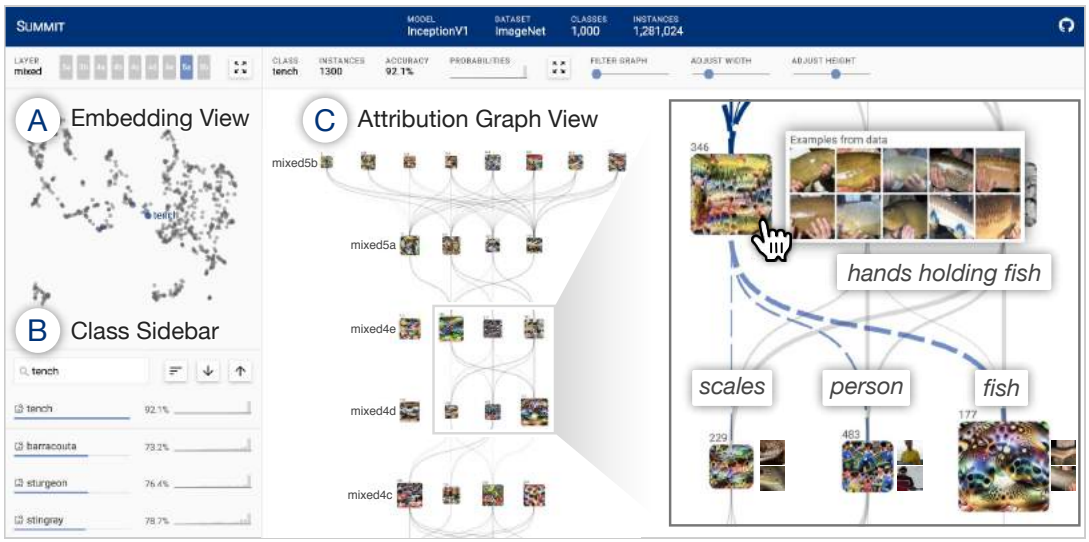


Figure 4.1: With Summit, users can scalably summarize and interactively interpret deep neural networks by visualizing *what* features a network detects and *how* they are related. In this example, InceptionV1 accurately classifies images of ***tench*** (yellow-brown fish). However, SUMMIT reveals surprising associations in the network (e.g., using parts of people) that contribute to its final outcome: the "tench" prediction is dependent on an intermediate "hands holding fish" feature (right callout), which is influenced by lower-level features like *"scales," "person,"* and *"fish"*. **(A) Embedding View** summarizes all classes' aggregated activations using dimensionality reduction. **(B) Class Sidebar** enables users to search, sort, and compare all classes within a model. **(C) Attribution Graph View** visualizes highly activated neurons as vertices ("scales," "fish") and their most influential connections as edges (dashed purple edges).

137, 72]. Our computational techniques for aggregating activations, aggregating influences, generating attribution graphs and their data, as well as the SUMMIT visualization, are open-sourced[1]. The system is available at the following public demo link: https://fredhohman.com/summit/.

- **Neural network exploration scenarios.** Using SUMMIT, we investigate how a widely-used computer vision model hierarchically builds its internal representation that has merely been illustrated in previous literature. We present neural network exploration scenarios where SUMMIT helps us discover multiple surprising insights into a prevalent, large-scale image classifier's learned representations and informs future neural network architecture design (Sec 4.6).
- **Broader impact for visualization in AI.** We believe our summarization approach that builds entire class representations is an important step for developing higher-level expla-

[1]Visualization: https://github.com/fredhohman/summit.
Code: https://github.com/fredhohman/summit-notebooks.
Data: https://github.com/fredhohman/summit-data.

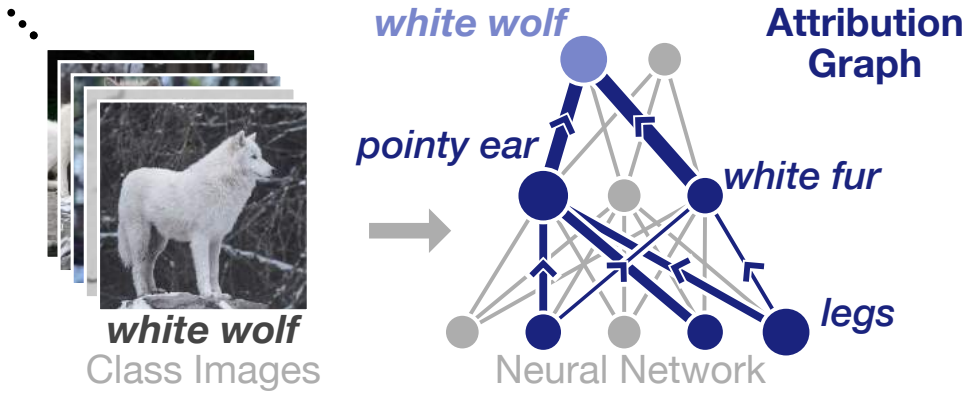


Figure 4.2: A high-level illustration of how we take thousands of images for a given class, e.g., images from ***white wolf*** class, compute their top activations and attributions, and combine them to form an **attribution graph** that shows how lower-level features ("legs") contribute to higher-level ones ("white fur"), and ultimately the final outcome.

nations for neural networks. We hope our work will inspire deeper engagement from both the information visualization and machine learning communities to further develop human-centered tools for artificial intelligence [136, 138].

4.1 Design Challenges

Our goal is to build an interactive visualization tool for users to better understand how neural networks build their hierarchical representation. To develop our summarization techniques and design SUMMIT, we identified five key challenges.

C1. SCALABILITY **Scaling up explanations and representations to entire classes, and ultimately, datasets of images.** Much of the existing work on interpreting neural networks focuses on visualizing the top independent activations or attributions for a single image [45, 43, 56, 100]. While this can be useful, it quickly becomes tiresome to inspect these explanations for more than a handful of images. Furthermore, since every image may contain different objects, to identify which concepts are representative of the learned model for a specific class, users must compare many image explanations together to manually find commonalities.

C2. INFLUENCE **Discovering influential connections in a network that most represents a learned class.** In dense neural network models, scalar edge weights directly connect neurons in a previous layer to neurons in a following layer; in other words, the activation of single neuron is expressed as a weighted sum of the activations from neurons in the previous layer [139]. However, this relationship is more complicated in convolutional neural networks. Images are convolved to form many 2D activation

maps, that are eventually summed together to form the next layers activations. Therefore, it becomes non-trivial to determine the effect of a single convolutional filter's effect on later layers.

C3. VISUALIZATION **Synthesizing meaningful, interpretable visualizations with important channels and influential connections.** Given a set of top activated neurons for a collection of images, and the impact convolutional filters have on later layers, how do we combine these approaches to form a holistic explanation that describes an entire class of images? Knowing how entire classes are represented inside of a model is important for trusting a model's predictions [26], aiding decision making in disease diagnosis [127, 128], devising security protocols [130], and fixing under-performing models [134, 26].

C4. INTERACTION **Interactive exploration of hundreds of learned class representations in a model.** How do we support interactive exploration and interpretation of hundreds or even thousands of classes learned by a prevalent, large-scale deep learning model? Can an interface support both high-level overviews of learned concepts in a network, while remaining flexible to support filtering and drilling down into specific features? Whereas **C1** focuses on the summarization approaches to scale up representations, this challenge focuses on interaction approaches for users to work with the summarized representations.

C5. RESEARCH ACCESS **High barrier of entry for understanding large-scale neural networks.** Currently, deep learning models require extensive computational resources and time to train and deploy. Can we make understanding neural networks more accessible without such resources, so that everyone has the opportunity to learn and interact with deep learning interpretability?

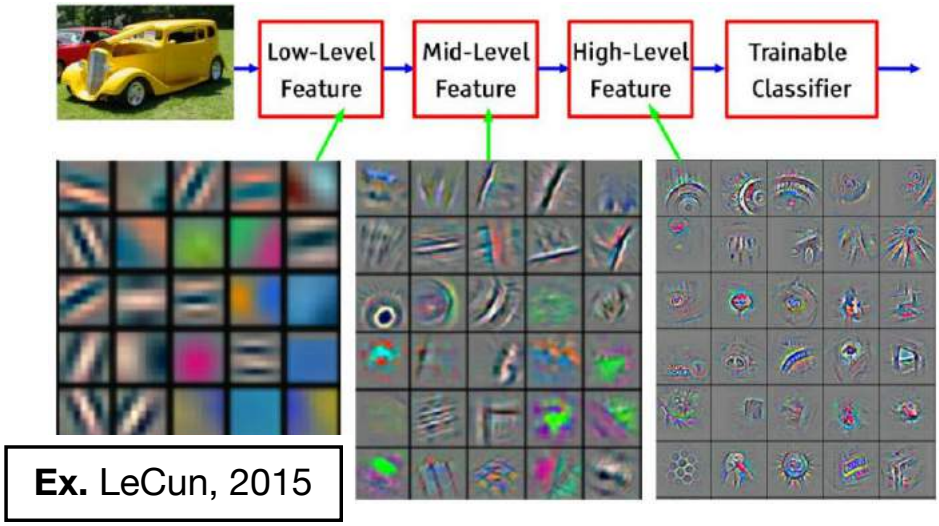


Figure 4.3: A common, widely shared example illustrating how neural networks learn hierarchical feature representations. Our work crystallizes these illustrations by systematically building a graph representation that describe *what* features a model has learned and *how* they are related. We visualize features learned at individual neurons and connect them to understand how high-level feature representations are formed from lower-level features. **Ex.** taken from Yann LeCun, 2015.

4.2 Design Goals

Based on the identified design challenges (Sec 4.1), we distill the following main design goals for SUMMIT, an interactive visualization system for summarizing what features a neural network has learned.

G1. Aggregating activations by counting top activated channels. Given the activations for an image, we can view them channel-wise, that is, a collection of 2D matrices where each encodes the magnitude of a detected feature by that channel's learned filter. We aim to identify which channels have the strongest activation for a given image, so that we can record only the topmost activated channels for every image, and visualize which channels, in aggregate, are most commonly firing a strong activation **(C1)**. This data could then be viewed as a feature of vector for each class, where the features are the counts of images that had a specific channel as a top channel (Sec 4.4.1).

G2. Aggregating influences by counting previous top influential channels. We aim to identify the most influential paths data takes throughout a network. If aggregated for every image, we could use intermediate outputs of the fundamental convolutional operation used inside of CNNs **(C2)** to help us determine which channels in a previous layer have the most impact on future channels for a given class of images (Sec

4.4.2).

G3. Finding what neural networks look for, and how they interact. To visualize how low-level concepts near early layers of a network combine to form high-level concepts towards later layers, we seek to form a graph from the entire neural network, using the aggregated influences as an edge list and aggregated activations as vertex values. With a graph representation, we could leverage the abundant research in graph algorithms, such as Personalized PageRank, to extract a subgraph that best captures the important vertices (neural network channels) and edges (influential paths) in the network (Sec 4.4.3). Attribution graphs would then describe the most activated channels and attributed paths between channels that ultimately lead the network to a final prediction **(C3)**.

G4. Interactive interface to visualize classes attribution graphs of a model. We aim to design and develop an interactive interface that can visualize entire attribution graphs (Sec 4.5). Our goal is to support users to freely inspect any class within a large neural network classifier to understand what features are learned and how they relate to one another to make predictions for any class **(C4)**. Here, we also want to use state-of-the-art deep learning visualization techniques, such as pairing feature visualization with dataset examples, to make channels more interpretable (Sec 4.5.3).

G5. Deployment using cross-platform, lightweight web technologies. To develop a visualization that is accessible for users without specialized computational resources, in SUMMIT we use modern web browsers to visualize attribution graphs (Sec 4.5). We also open-source our code to support reproducible research **(C5)**.

4.3 Model Choice and Background

In this work, we demonstrate our approach on InceptionV1 [27], a prevalent, large-scale convolutional neural network (CNN) that achieves top-5 accuracy of 89.5% on the ImageNet dataset that contains over 1.2 millions images across 1000 classes. InceptionV1 is composed of multiple inception modules: self-contained groups of parallel convolutional layers. The last layer of each inception module is given a name of the form "mixed{number}{letter}," where the {number} and {letter} denote the location of a layer in the network; for example, mixed3b (an earlier layer) or mixed4e (a later layer). In InceptionV1, there are 9 such layers: mixed3{a,b}, mixed4{a,b,c,d,e}, and mixed5{a,b}. While there are more technical complexities regarding neural network design within each inception module, we follow existing interpretability literature and consider the 9 mixed layers as the primary layers of the network [56, 100]. Although our work makes this model choice, our proposed summarization and visualization techniques can be applied to other neural network architectures in other domains.

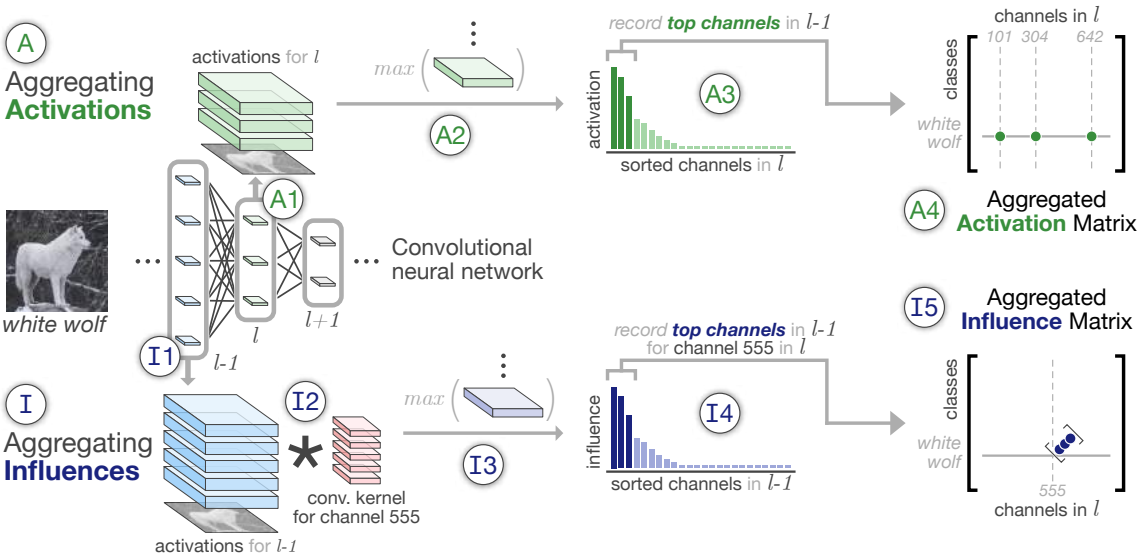


Figure 4.4: Our approach for aggregating activations and influences for a layer l. **Aggregating Activations**: **(A1)** given activations at layer l, **(A2)** compute the max of each 2D channel, and **(A3)** record the top activated channels into an **(A4)** aggregated activation matrix, which tells us which channels in a layer most activate and represent every class in the model. **Aggregating Influences**: **(I1)** given activations at layer $l-1$, **(I2)** convolve them with a convolutional kernel from layer l, **(I3)** compute the max of each resulting 2D activation map, and **(I4)** record the top most influential channels from layer $l-1$ that impact channels in layer l into an **(I5)** aggregated influence matrix, which tells us which channels in the previous layer most influence a particular channel in the next layer.

4.4 Creating Attribution Graphs by Aggregation

SUMMIT introduces two new scalable summarization techniques: (1) *activation aggregation* discovers important neurons, and (2) *neuron-influence aggregation* identifies relationships among such neurons. SUMMIT combines these techniques to create the novel *attribution graph* that reveals and summarizes crucial neuron associations and substructures that contribute to a model's outcomes. Attribution graphs tell us *what* features a neural network detects, and *how* those features are related. Below, we formulate each technique, and describe how we combine them to generate attribution graphs (Sec 4.4.3) for CNNs.

4.4.1 Aggregating Neural Network Activations

We want to understand *what* a neural network is detecting in a dataset. We propose summarizing how an image dataset is represented throughout a CNN by aggregating individual image **activations** at each channel in the network, over all of the images in a given class. This aggregation results in a matrix, A^l for each layer l in a network, where an entry A^l_{cj} roughly represents how *important* channel j (from the l^{th} layer) is for representing im-

ages from class c. This measure of importance can be defined in multiple ways, which we discuss formally below.

A convolutional layer contains C_l image kernels (parameters) that are convolved with an input image, X, to produce an output image, Y, that contains C_l corresponding channels. For simplicity, we assume that the hyperparameters of the convolutional layer are such that X and Y will have the same height H and width W, i.e., $X \in \mathbb{R}^{H \times W \times C_{l-1}}$ and $Y \in \mathbb{R}^{H \times W \times C_l}$. Each channel in Y is a matrix of values that represent how strongly the corresponding kernel *activated* in each spatial position. For example, an edge detector kernel will produce a channel, also called an activation map, that has larger values at locations where an edge is present in the input image. As kernels in convolutional layers are learned during model training, they identify different features that discriminate between different image classes. It is commonly thought that CNNs build hierarchical feature representations of input images, learning simple edge and shape detectors in early layers of the network, which are combined to form texture detectors, and finally relevant object detectors in later layers of the network [52] (see Fig 4.3).

A decision must be made on how to aggregate activations over spatial locations in a channel and aggregate activations over all images in a given class. Ultimately, we want to determine channel importance in a CNN's representation of a class. As channels roughly represent concepts, we choose the maximum value of a channel as an indicator of how strongly a concept is present, instead of other functions, such as mean, that may dampen the magnitude of relevant channels.

Alongside Fig 4.4, our method for aggregation is as follows:

- **Compute activation channel maximums for all images.** For each image, **(A1)** obtain its activations for a given layer l and **(A2)** compute the maximum value per channel. This is equivalent to performing Global Max-pooling at each layer in the network. Now for each layer, we will have a matrix Z^l, where an entry Z^l_{ij} represents the maximum activation of image i over the j^{th} channel in layer l.
- **Filter by a particular class.** We consider all rows of Z^l whose images belong to the same class, and want to aggregate the maximum activations from these rows to determine which channels are important for detecting the class.
- **Aggregation Method 1: taking top k_{M1} channels.** For each row, we set the top k_{M1} largest elements to 1 and others to 0, then sum over rows. Performing this operation for each class in our dataset will result in a matrix A^l from above where an entry A^l_{cj} is the count of the number of times that the j^{th} channel is one of the top k_{M1} channels by maximum activation for all images in class c. This method ignores the actual maximum activation values, so it will not properly handle cases where a single channel activates strongly for images of a given class (as it will consider $k_{M1} - 1$ other channels), or cases where many channels are similarly activated over images of a given class (as it will *only* consider k_{M1} channels as "important"). This observation motivates our second method.

- **Aggregation Method 2: taking top $k_{M2}\%$ of channels by weight.** We first scale rows of Z^l to sum to 1 by dividing by the row sums, $Z'^l_{ij} = \frac{Z^l_{ij}}{\sum_{n=1}^{N} Z^l_{nj}}$, where N is the number of images. Instead of setting the top k_{M2} elements to 1, as in **Method 1**, we set the m largest elements of each row to 1 and the remaining to 0. Here, m is the largest index such that $\sum_{j \in \text{sorted } Z'^l_i}^{m} Z'^l_{ij} \leq k_{M2}$, where k_{M2} is some small percentage. In words, this method first sorts all channels by their maximum activations, then records channels, starting from the largest activated, until the cumulative sum of probability weight from the recorded channels exceeds the threshold. Contrary to **Method 1**, this method adaptively chooses channels that are important for representing a given image, producing a better final class representation.

Empirically, we noticed the histograms of max channel activations was often power law distributed, therefore we use **Method 2** to **(A3)** record the top $k_{M2} = 3\%$ of channels to include in the **(A4) Aggregated Activations** matrix A^l. In terms of runtime, this process requires only a forward pass through the network.

4.4.2 Aggregating Inter-layer Influences

Aggregating activations at each convolutional layer in a network will only give a local description of which channels are important for each class, i.e., from examining A^l we will not know *how* certain channels come to be the most representative for a given class. Thus, we need a way to calculate how the activations from the channels of a previous layer, $l-1$, **influence** the activations at the current layer, l. In dense layers, this influence is trivial to compute: the activation at a neuron in l is computed as the weighted sum of activations from neurons in $l-1$. The influence of a single neuron from $l-1$ is then proportional to the activation of that neuron multiplied by the associated weight to the neuron being examined from l. In convolutional layers, calculating this influence is more complicated: the activations at a channel in l are computed as the 3D convolution of all of the channels from $l-1$ with a learned kernel tensor. This operation can be broken down (shown formally later in this section) as a summation of the 2D convolutions of each channel in $l-1$ with a corresponding slice of the appropriate kernel. The summations of 2D convolutions are similar in structure to the weighted-summations performed by dense layers, however the corresponding "influence" of a single channel from $l-1$ on the output of a particular channel in l is a 2D feature map. We can summarize this feature map into a scalar influence value by using any type of reduce operation, which we discuss further below.

We propose a method for (1) quantifying the *influence* a channel from a previous layer has on the activations of a channel in a following layer, and (2) aggregating influences into a tensor, I^l, that can be interpreted similarly to the A^l matrix from the previous section. Formally, we want to create a tensor I^l for every layer l in a network, where an entry I^l_{cij} represents how important channel i from layer $l-1$ is in determining the output of channel j in layer l, for all images in class c.

First, using the notation from the previous section, we consider how a single channel of Y is created from the channels of X. Let $K^{(j)} \in \mathbb{R}^{H \times W \times C_{l-1}}$ be the j^{th} kernel of our convolutional layer. Now the operation of a convolutional layer can be written as:

$$Y_{:,:,j} = \underbrace{X * K^{(j)}}_{\text{3D convolution}} = \sum_{i=1}^{C_{l-1}} \underbrace{X_{:,:,i} * K^{(j)}_{:,:,i}}_{\text{2D convolution}} \tag{4.1}$$

In words, **(I1)** each channel from X is **(I2)** convolved with a slice of the j^{th} kernel, and the resulting maps are summed to produce a single channel in Y. We care about the 2D quantity $X_{:,:,i} * K^{(j)}_{:,:,i}$ as it contains exactly the contributions of a *single* channel from the previous layer to a channel in the current layer.

Second, we must summarize the quantity $X_{:,:,i} * K^{(j)}_{:,:,i}$ into a scalar influence value. Similarly discussed in Sec 4.4.1, this can be done in many ways, e.g., by summing all values, applying the Frobenius norm, or taking the maximum value. Each of these summarization methods (i.e., 2D to 1D reduce operations) may lend itself well to exposing interesting connections between channels later in our pipeline. We chose to **(I3)** take the maximum value of $X_{:,:,i} * K^{(j)}_{:,:,i}$ as our measure of influence for the image classification task, since this task intuitively considers the largest magnitude of a feature, e.g., how strongly a "dog ear" or "car wheel" feature is expressed, instead of summing values for example, which might indicate how many places in the image a "dog ear" or "car wheel" is being expressed. Also, this mirrors our approach for aggregating activations above.

Lastly, we must aggregate these influence values between channel pairs in consecutive layers, for all images in a given class, i.e., create the proposed I^l matrix from the pairwise channel influence values. This process mirrors the aggregation described previously (Sec 4.4.1), and we follow the same framework. Let L^l_{ij} be the scalar influence value computed by the previous step *for a single image in class* c, between channel i in layer $l-1$ and channel j in layer l. We increment an entry (c, i, j) in the tensor I^l_{cij} if L^l_{ij} is one of the top k_{M1} largest values in the column $L^l_{:,j}$ (mirroring **Method 1** from Sec 4.4.2), or if L^l_{ij} is in the top $k_{M2}\%$ of largest values in $L^l_{:,j}$ (mirroring **Method 2** (Sec 4.4.1).

Empirically, we noticed the histograms of max influence values were not as often power law distributed as in the previous aggregation of activations, therefore we use **Method 1** to **(I4)** record the top $k_{M1} = 5$ channels to include in the **(I5) Aggregated Influence** matrix I^l. Note that InceptionV1 contains inception modules, groups of branching parallel convolution layers. Our influence aggregation approach handles these layer depth imbalances by merging paths using the minimum of any two hop edges through an inner layer; this guarantees all edge weights between two hop channels are maximal. In terms of runtime, this process is more computationally expensive than aggregating activations, since we have to compute all intermediate 2D activation maps; however, with a standard GPU equipped machine is sufficient. We discuss our experimental setup later in Sec 4.5.4.

4.4.3 Combining Aggregated Activations and Influences to Generate Attribution Graphs

Given the aggregated activations A^l and aggregated influences I^l we aim to combine them into a single entity that describes both *what* features a neural network is detecting and *how* those features are related. We call these **attribution graphs**, and we describe their generation below.

In essence, neural networks are directed acyclic graphs: they take input data, compute transformations of that data at sequential layers in the network, and ultimately produce an output. We can leverage this graph structure for our desired representation. Whereas a common network graph has vertices and connecting edges, our vertices will be the channels of a network (for all layers of the network), and edges connect channels if the channel in the previous layer has a strong influence to a channel in an later, adjacent layer.

Using graph algorithms for neural network interpretability. Consider the aggregated influences I^l as an edge list; therefore, we can build an "entire graph" of a neural network, where edges encode if an image had a path from one channel to another as a top influential path, and the weight of an edge is a count of the number of images for a given class with that path as a top influential path. Now, for a given class, we want to extract the subgraph that best captures the important vertices (channels) and edges (influential paths) in the network. Since we have instantiated a typical network graph, we can now leverage the abundant research in graph algorithms. A natural fit for our task is the Personalized PageRank algorithm [140], which scores each vertex's importance in a graph, based on both the graph structure and the weights associated with the graph's vertices and edges. Specifically, SUMMIT operates on the graph produced from all the images of a given class; the algorithm is initialized by and incorporates both vertex information (aggregated activations A^l) and edge information (aggregated influences I^l) to find a subgraph most relevant for all the provided images. We normalize each layer's personalization from A^l by dividing by max A^l value for each layer l so that each layer has a PageRank personalization within 0 to 1. This is required since each layer has a different total number of possible connections (e.g., the first and last layers, mixed3a and mixed5b, only have one adjacent layer, therefore their PageRank values would be biased small). In summary, we make the full graph of a neural network where vertices are channels from all layers in the network with a personalization from A^l, and edges are influences with weights from I^l.

Extracting attribution graphs. After running Personalized PageRank for 100 iterations, the last task is to select vertices based on their computed PageRank values to extract an attribution graph. There are many different ways to do this; below we detail our approach. We first compute histograms of the PageRank vertex values for each layer. Next, we use the methodology described in Sec 4.4.1 for **Method 2**, where we continue picking vertices with the largest PageRank value until we have reached $k_{M2}\%$ weight for each layer independently. Empirically, here we set $k_{M2} = 7.5\%$ after observing that the PageRank value histograms are roughly power law, indicating that there are only a handful of

channels determined important. Regarding the runtime, the only relevant computation is running PageRank on the full neural network graph, which typically has a few thousands vertices and a few hundred thousand edges. Using the Python NetworkX[2] implementation [140], Personalized PageRank runs in about 30 seconds for each class.

4.5 The SUMMIT User Interface

From our design goals in Sec 4.2 and our aggregation methodology in Sec 4.4, we present SUMMIT, an interactive system for scalable summarization and interpretation for exploring entire learned classes in large-scale image classifier models (Fig 4.1).

The header of SUMMIT displays metadata about the visualized image classifier, such as the model and dataset name, the number of classes, and the total number data instances within the dataset. As described in Sec 4.3, here we are using InceptionV1 trained on the 1.2 million image dataset ImageNet that contains 1000 classes. Beyond the header, the SUMMIT user interface is composed of three main interactive views: the Embedding View, the Class Sidebar, and the Attribution Graph View. The following section details the representation and features of each view and how they tightly interact with one another.

4.5.1 Embedding View: Learned Class Overview

The first view of SUMMIT is the Embedding View, a dimensionality reduction overview of all the classes in a model (Fig 4.1A). Given some layer l's A^l matrix, recall an entry in this matrix corresponds to the number of images from one class (row) that had one channel (column) as a top channel. We can consider A as a feature matrix for each class where the number of channels in a layer corresponds to the number of features. For reduction and visualization, the Embedding View uses UMAP: a non-linear dimensionality reduction that better preserves global data structure, compared to other techniques like t-SNE, and often provides a better "big picture" view of high-dimensional data while preserving local neighbor relations [111]. Each dot corresponds to one class of the model, with spatial position encoding their similarity. To explore this embedding, users can freely zoom and pan in the view, and when a user zooms in close enough, labels appear to describe each class (point) so users can easily see how classes within the model compare. Clicking on a point in the Embedding View will update the selection for the remaining views of SUMMIT, as described below.

Selectable neural network minimap. At the top of the Embedding View sits a small visual representation of the considered neural network; in this case, InceptionV1's primary mixed layers are shown (Fig 4.5). Since we obtain one A^l matrix for every layer l in the model, to see how the classes related to one another at different layer depths within the

[2]NetworkX: https://networkx.github.io/

network, users can click on one of the other layers to animate the Embedding View. This is useful for obtaining model debugging hints and observing at a high-level how classes are represented throughout a network's layers.

Figure 4.5: Selectable network minimap animates the Embedding View.

4.5.2 Class Sidebar: Searching and Sorting Classes

Underneath the Embedding View sits the Class Sidebar (Fig 4.1B): a scrollable list of all the class of the model, containing high-level class performance statistics. The first class at the top of the list is the selected class, whose attribution graph is shown in the Attribution Graph View, to be discussed in the next section. The Class Sidebar is sorted by the similarity of the selected class to all other classes in the model. For the similarity metric, we compute the cosine similarity using the values from A^l. Each class is represented as a horizontal bar that contains the class's name, a purple colored bar that indicates its similarity to the selected class (longer purple bars indicate similar classes, and vice versa), the class's top-1 accuracy for classification, and a small histogram of all the images' predicted probabilities within that class (i.e., the output probabilities from the final layer) (Fig 4.6). From this small histogram, users can quickly see how well a class performs. For example, classes with power law histograms indicate high accuracy, whereas classes with normal distribution histograms indicate underperformance. Users can then hypothesize whether a model may be biasing particular classes over others, or if underperforming classes have problems with their raw data.

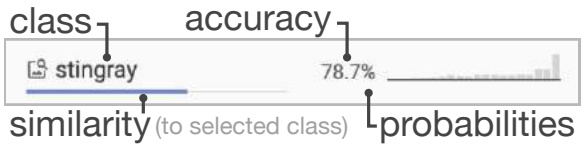


Figure 4.6: Class Sidebar visual encoding.

Scrolling for context. To see where a particular class in the sidebar is located in the Embedding View, users can hover over a class to highlight its point and label the Embedding View above (Fig 4.1A-B). Since the Class Sidebar is sorted by class similarity, to see where similar classes lie compared to the selected class, all classes in the Class Sidebar visible to the user (more technically, in the viewbox of the interface) are also highlighted in the Embedding View (Fig 4.1A-B). Scrolling then enables users to quickly see where classes in the Class Sidebar lie in the Embedding View as classes become less similar to the originally selected class to visualize.

Sorting and selecting classes. To select a new class to visualize, users can click on any class in the Class Sidebar to update the interface, including resorting the Class Sidebar by similarity based on the newly selected class and visualize the new class's attribution graph in the Attribution Graph View. Users can also use the search bar to directly search for a known class instead of freely browsing the Class Sidebar and Embedding View. Lastly, the Class Sidebar has two additional sorting criteria. Users can sort the Class Sidebar by the accuracy, either ascending or descending, to see which classes in the model have the highest and lowest predicted accuracy, providing a direct mechanism to begin to inspect and debug underperforming classes.

4.5.3 Attribution Graph View: Visual Class Summarization

The Attribution Graph View is the main view of SUMMIT (Fig 4.1C). A small header on top displays some information about the class, similar to that in the Class Sidebar, and contains a few controls for interacting with the attribution graph, to be described later.

Visualizing attribution graphs. Recall from Autorefsubsec:combine that an attribution graph is a subgraph of the entire neural network, where the vertices correspond to a class's important channels within a layer, and the edges connect channels based on their influence from the convolution operation. Our graph visualization design draws inspiration from recent visualization works, such as CNNVis [141], AEVis [69], and Building Blocks [100], that have successfully leveraged graph based representations for deep learning interpretability. In the main view of SUMMIT, an attribution graph is shown in a zoomable and panable canvas that visualizes the graph vertically, where the top corresponds to the last mixed network layer in the network, mixed5b, and the bottom layer corresponds to the first mixed layer, mixed3a (Fig 4.1C). In essence, the attribution graph is a directed network with vertices and edges; in SUMMIT, we replace vertices with the corresponding channel's feature visualization. Each layer, denoted by a label, is a horizontal row of feature visualizations of the attribution graph. Each feature visualization is scaled by its magnitude of the number of images within that class that had that channel as a top channel in their prediction, i.e., the value from A^l. Edges are drawn connecting each channel to visualize the important paths data takes during prediction. Edge thickness is encoded by the influence from one channel to another, i.e., the value from I^l.

Understanding attribution graph structure. This novel visualization reveals a number of interesting characteristics about how classes behave inside a model. First, it shows how neural networks build up high-level concepts from low-level features, for example, in the ***white wolf*** class, early layers learn fur textures, ear detectors, and eye detectors, which all contribute to form face and body detectors in later layers. Second, the number of visualized channels per layer roughly indicates how many features are needed to represent that class within the network. For example, in layer mixed5a, the ***strawberry*** class only has a few large channels, indicating this layer has learned specific object detectors for

strawberries already, whereas in the same layer, the ***drum*** class has many smaller channels, indicating that this layer requires the combination of multiple object detectors working together to represent the class. Third, users can also see the overall structure of the attribution graph, and how a model has very few important channels in earlier layers, but as the the network progress, certain channels grow in size and begin to learn high-level features about what an image contains.

Inspecting channels and connections in attribution graphs. Besides displaying the feature visualization at each vertex, there are a number of different complementary data that is visualized to help interpret what a model has learned for a given class attribution graph. It has been shown that for interpreting channels in a neural network, feature visualization is not always enough [56]; however, displaying example image patches from the entire dataset next to a feature visualization helps people better understand what the channel is detecting. We apply a similar approach, where hovering over a channel reveals 10 image patches from the entire dataset that most maximize this specific channel (Fig 4.1C). Pairing feature visualization with dataset examples helps understand what the channel is detecting in the case where a feature visualization alone is hard to decipher. When a user hovers over a channel, SUMMIT also highlights the edges that flow in and out of that specific channel by coloring the edges and animating them within the attribution graph. This is helpful for understanding which and how much channels in a previous layer contribute to a new channel in a later layer. Users can also hover over the edges of an attribution graph to color and animate that specific edge and its endpoint channels, similar to the interaction used when hovering over channels. Lastly, users can get more insight into what feature a specific channel has learned by hovering left to right on a channel to see the feature visualization change to display four other feature visualizations generated with *diversity*: a technique used to create multiple feature visualizations for a specific channel at once that reveals different areas of latent space that a channel has learned [56]. This interaction is inspired from commercial photo management applications where users can simply hover over an image album's thumbnail to quickly preview what images are are inside.

Dynamic drill down and filtering. When exploring an attribution graph, users can freely zoom and pan the entire canvas, and return to the zoomed-out overview of the visualization via a button included in the options bar above the attribution graph. In the case of a large attribution graph where there are too many channels and edges, in the options bar there is a slider that when dragged, filters the the channels of the attribution graph by their importance from A^l. This interaction technique draws inspiration from existing degree-of-interest graph exploration research, where users can dynamically filter and highlight a subset of the most important channels (vertices) and connections (edges) based on computed scores [142, 143, 144, 145]. Dragging the slider triggers an animation where the filtered-out channels and their edges are removed from the attribution graph, and the remaining visualization centers itself for each layer. With the additional width and height sliders, these interactions add dynamism to the attribution graph, where it fluidly animates

and updates to users deciding the scale of the visualization.

4.5.4 System Design

To broaden access to our work, SUMMIT is web-based and can be accessed from any modern web-browser. SUMMIT uses the standard HTML/CSS/JavaScript stack, and D3.js[3] for rendering SVGs. We ran all our deep learning code on a NVIDIA DGX 1, a workstation with 8 GPUs, with 32GB of RAM each, 80 CPU cores, and 504GB of RAM. With this machine we could generate everything required for *all 1000 ImageNet* classes—aggregating activations, aggregating influences, and combining them with PageRank (implementation from NetworkX) to form attribution graphs—and perform post-processing under 24 hours. However, visualizing a single class on one GPU takes only a few minutes. The *Lucid* library is used for creating feature visualizations[4], and dataset examples are used from the appendix[5] of [56].

4.6 Neural Network Exploration Scenarios

4.6.1 Unexpected Semantics Within a Class

A problem with deploying neural networks in critical domains is their lack of interpretability, specifically, can model developers be confident that their network has learned what they think it has learned? We can answer perplexing questions like these with SUMMIT. For example, in Fig 4.1, consider the ***tench*** class (a type of yellow-brown fish). Starting from the first layer, as we explore the attribution graph for ***tench*** we notice there are no fish or water feature, but there are many "finger", "hand", and "people" detectors. It is not until a middle layer, mixed4d, that the first fish and scale detectors are seen (Fig 4.1C, callout); however, even these detectors focus solely on the body of the fish (there is no fish eye, face, or fin detectors). Inspecting dataset examples reveals many image patches where we see people's fingers holding fish, presumably after catching them. This prompted us to inspect the raw data for the ***tench*** class, where indeed, most of the images are of a person holding the fish. We conclude that, unexpectedly, the model uses people detectors and in combination with brown fish body and scale detectors to represent the ***tench*** class. Generally, we would not expect "people" as an essential feature for classifying fish.

This surprising finding motivated us to seek another class of fish that people do not normally hold to compare against, such as a ***lionfish*** (due to their venomous spiky fin rays). Visualizing the ***lionfish*** attribution graph confirms our suspicion (Fig 4.7): there are

[3]D3.js: https://d3js.org/
[4]Lucid: https://github.com/tensorflow/lucid
[5]https://github.com/distillpub/post--feature-visualization

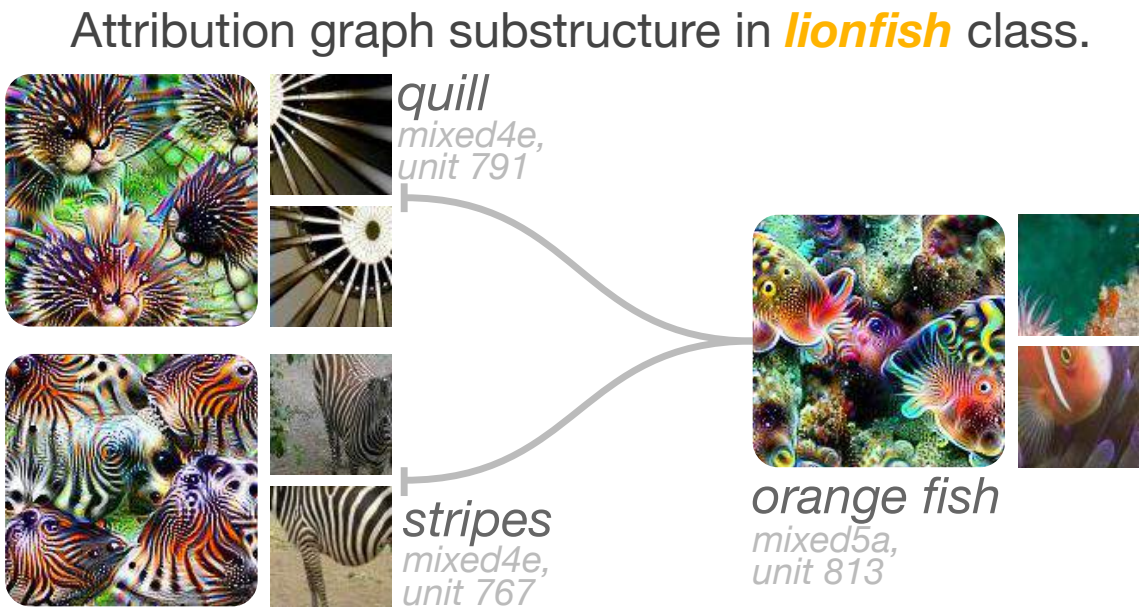


Figure 4.7: An example substructure of the ***lionfish*** attribution graph reveals unexpected texture features such as "quills" and "stripes," which significantly influence the most activated channels in a final layer responsible for identifying the "orange fish" feature. It is noteworthy that some ***lionfish*** species are reddish-orange, and have white fin rays.

not any people object detectors in its attribution graph. However, we discover yet another unexpected combination of features: there are few fish part detectors while there are many texture features, e.g., stripes and quills. It is not until the final layers of the network where a highly activated channel detects orange fish in water, which uses the stripe and quill detectors. Therefore we deduce that the ***lionfish*** class is composed of a striped body in the water with long, thin quills. Whereas the ***tench*** had unexpected people features, the ***lionfish*** lacked fish features. Regardless, findings such as these can help people more confidently deploy models when they know what composition of features results in a prediction.

4.6.2 Mixed Class Association Throughout Layers

While inspecting the Embedding View, we noticed some classes' embedding positions shift greatly between adjacent layers. This cross-layer embedding comparison is possible since each layer's embedding uses the previous layer's embedding as an initialization. Upon inspection, the classes that changed the most were classes that were either a combination of existing classes or had *mixed primary associations*.

For example, consider the ***horsecart*** class. For each layer, we can inspect the nearest neighbors of ***horsecart*** to check its similarity to other classes. We find that ***horsecart*** in the early layers is similar to other **mechanical** classes, e.g., harvester, thresher, and snowplow. This association shifts in the middle layers where ***horsecart*** moves to be near **animal**

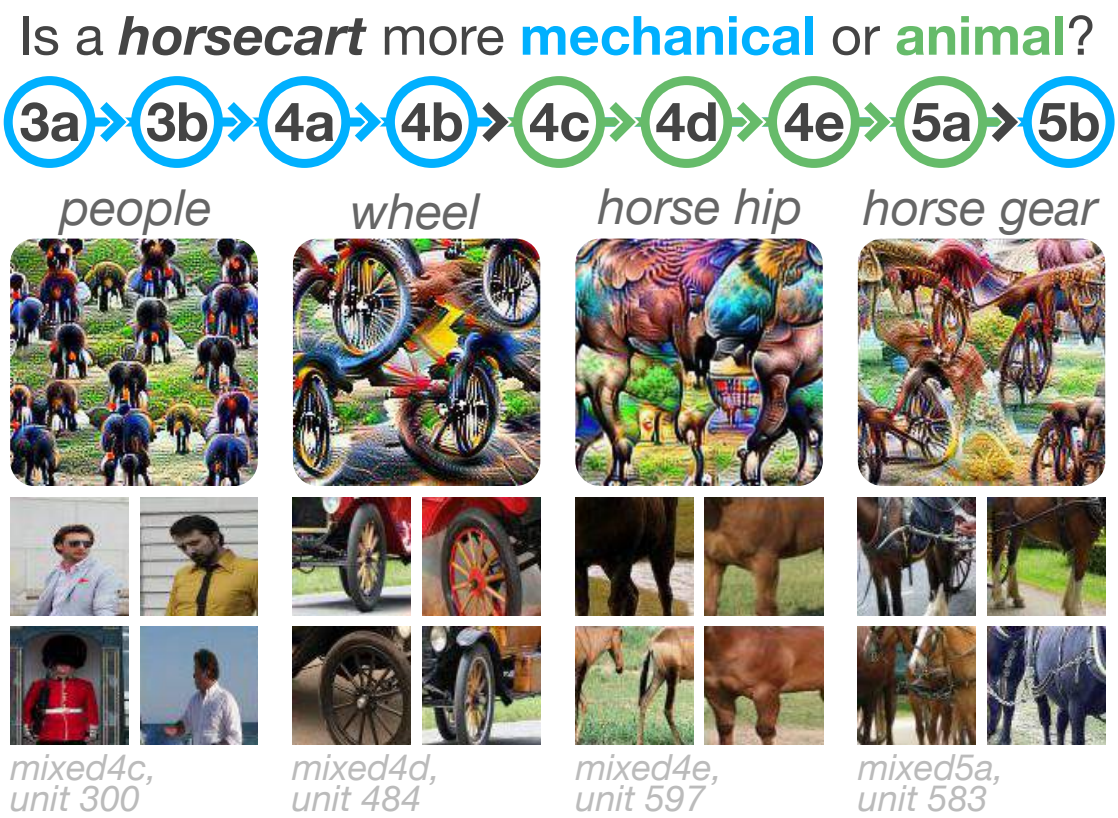


Figure 4.8: Using SUMMIT we can find classes with mixed semantics that shift their primary associations throughout the network layers. For example, early in the network, ***horsecart*** is most similar to **mechanical** classes (e.g., harvester, thresher, snowplow), towards the middle it shifts to be nearer to **animal** classes (e.g., bison, wild boar, ox), but ultimately returns to have a stronger **mechanical** association at the network output.

classes, e.g., bison, wild boar, and ox. However, ***horsecart*** flips back at the final convolutional layer, returning to a **mechanical** association (Fig 4.8, top). To better understand what features compose a ***horsecart***, we inspect its attribution graph and find multiple features throughout all the layers that contain people, spoke wheels, horse hips, and eventually horse bodies with saddles and mechanical gear (Fig 4.8, bottom). Mixed semantic classes like ***horsecart*** allow us to test if certain classes are semantic combinations of others and probe deeper into understanding how neural networks build hierarchical representations.

4.6.3 Discriminable Features in Similar Classes

Since neural networks are loosely inspired by the human brain, in the broader machine learning literature there is great interest to understand if decision rationale in neural networks is similar to that of humans. With attribution graphs, we can further to answer this question by comparing classes throughout layers of a network.

For example, consider the ***black bear*** and ***brown bear*** classes. A human would likely

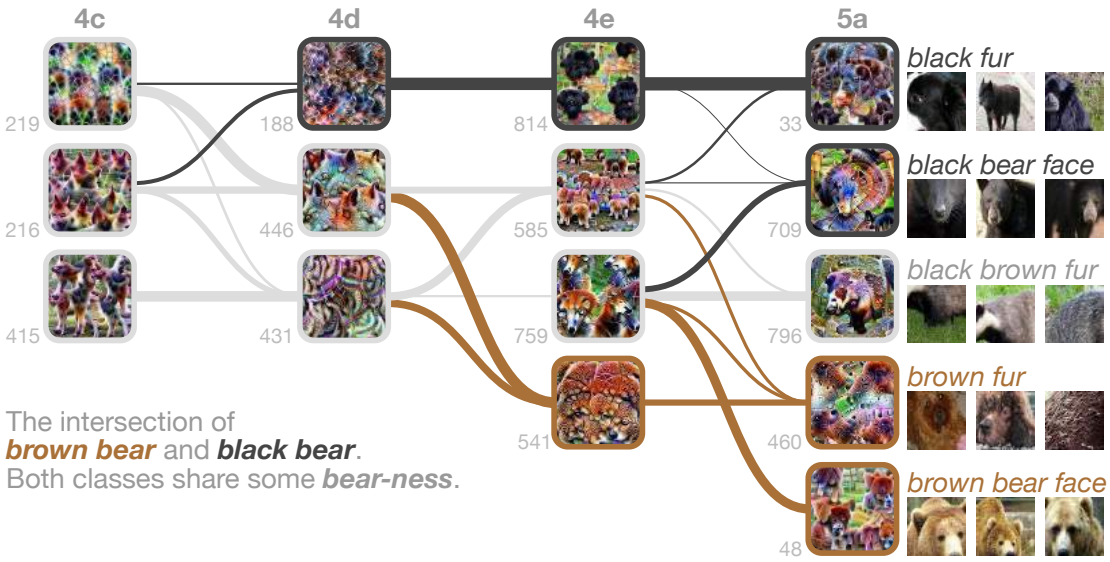


Figure 4.9: With attribution graphs, we can compare classes throughout layers of a network. Here we compare two similar classes: ***black bear*** and ***brown bear***. From the intersection of their attribution graphs, we see both classes share features related to ***bear-ness***, but diverge towards the end of the network using fur color and face color as discriminable features. This feature discrimination aligns with how humans might classify bears.

say that color is the discriminating difference between these classes. By taking the *intersection* of their attribution graphs, we can see what features are shared between the classes, as well as any discriminable features and connections. In Fig 4.9, we see in earlier layers (mixed4c) that both ***black bear*** and ***brown bear*** share many features, but as we move towards the output, we see multiple diverging paths and channels that distinguish features for each class. Ultimately, we see individual black and brown fur and bear face detectors, while some channels represent general ***bear-ness***. Therefore, it appears InceptionV1 classifies ***black bear*** and ***brown bear*** based on color, which may be the primary feature humans may classify by. This is only one example, and it is likely that these discriminable features do not always align with what we would expect; however, attribution graphs give us a mechanism to test hypotheses like these.

4.6.4 Finding Non-semantic Channels

Using SUMMIT, we quickly found several channels that detected non-semantic, irrelevant features, regardless of input image or class (verified manually with 100+ classes, computationally with all). For example, in layer mixed3a, channel 67 activates to the image frame, as seen in Fig 4.10. We found 5 total non-semantic channels, including mixed3a 67, mixed3a 190, mixed3b 390, mixed3b 399, and mixed3b 412. Upon finding these, we reran our algorithm for aggregating activations and influences, and generated all attribution graphs with these channels excluded from the computation, since they consistently

Figure 4.10: Using SUMMIT on InceptionV1 we found non-semantic channels that detect irrelevant features, regardless of the input image, e.g., in layer mixed3a, channel 67 is activated by the frame of an image.

produced high activation values but were incorrectly indicating important features in many classes. Although SUMMIT leverages recent feature visualization research [56] to visualize channels, it does not provide an automated way to measure the semantic quality of channels. We point readers to the appendix of [56] to explore this important future research direction.

4.6.5 Informing Future Algorithm Design

We noticed that some classes (e.g., ***zebra***, ***green mamba***) have only a few important channels in the middle layers of the network, indicating that these channels could have enough information to act as a predictor for the given class. This observation implies that it may be prudent to make classification decisions at different points in the network, as opposed to after a single softmax layer at the output. More specifically, per the A^l matrices, we can easily find these channels (in all layers) that maximally activates for each class. We could then perform a MaxPooling operation at each of these channels, followed by a Dense layer classifier to form a new "model" that only uses the most relevant features for each class to make a decision.

The inspiration for this proposed algorithm is a direct result of the observations made possible by SUMMIT. Furthermore, our proposed methodology makes it easy to test whether the motivating observation holds true for other networks besides InceptionV1. It could be the case that single important channels for certain classes are a result of the training with multiple softmax 'heads' used by InceptionV1; however, without SUMMIT, checking this would be difficult.

4.7 Discussion and Future Work

Interactive visual comparison of attribution graphs. Currently, SUMMIT interactively visualizes single attribution graphs. However, there is great opportunity to support automatically, visual comparison between multiple attribution graphs. Example comparison operations include computing attribution graph difference, union, and intersection.

Mining attribution graphs for subgraph motifs. Since attribution graphs are regular network graphs, we can leverage data mining and graph analysis techniques to find the most common motifs, e.g., all mammal classes may have three specific channels that form a triangle that is always activated highly, or maybe all car classes share only single path throughout the network. Extracting these smaller subgraph motifs could give deep insight into how neural networks arrange hierarchical concepts inside their internal structure.

Visualizing other neural network models. We justify our model choice in Sec 4.3, but an immediate avenue for future work explores generating attribution graphs on other CNN models. Simpler models like VGG [33] can be easily adapted with our approach, but more complex networks like ResNets [119] will require a small modification for computing attribution and influences (e.g., considering skip connections between layers as additional graph edges). Our approach also may be adopted for exploring neural network components of model architectures that provide activation information (e.g., the two individual networks within a GAN [146], but not their interaction).

Better attribution graph generation. Computing neural network attribution remains an active area of research: there is no consensus of the best way to compute attribution [100, 45, 147, 148, 149, 52, 42]. To generate attribution graphs, we use activation aggregation as an initialization for personalized PageRank on the entire network from aggregated influences. While this is one effective way to generate attribution graphs, there could be other ways to generate graph explanations that describe learned neural network representations. If so, this will only improve the value of SUMMIT's visualizations. For example, layer-wise relevance propagation [150] could be used to seed our aggregation methods using relevance scores instead of neuron activations. Conversely, exploring attribution graphs using less-contributing channels could be a novel way to discover non-relevant features. However, aggregation over spatial positions and instances, a main contribution of SUMMIT, will still be necessary given any other measure of neuron importance.

Hyperparameter selections. Our approach has a few hyperparameters choices, including determining how many channels to record per image when aggregating activations and computing attribution graph influences, as well as what PageRank threshold to set for creating the final visualizations. However, since our approach was designed to take advantage of data at scale, in our tests we do not see many differences in the limit that the number of images increases. Note that while our approach benefits from scale, both the aggregation and visualization work on arbitrary dataset sizes, e.g., a single image, hundreds, or

thousands.

Longitudinal evaluation of impacts in practice. We presented Summit to ML researchers and scientists at industry and government research labs, and discussed plans to conduct long-term studies to test Summit on their own models. We plan to investigate how Summit may inform algorithmic model design, prompt data collection for ill-represented classes, and discover latent properties of deployed models.

4.8 Conclusion

As deep learning is increasingly used in decision-making tasks, it is important to understand how neural networks learn their internal representations of large datasets. In this work, we present SUMMIT, an interactive system that scalably and systematically summarizes and visualizes what features a deep learning model has learned and how those features interact to make predictions. The SUMMIT visualization runs in modern web browsers and is open-sourced. We believe our summarization approach that builds entire class representations is an important step for developing higher-level explanations for neural networks. We hope our work will inspire deeper engagement from both the information visualization and machine learning communities to further develop human-centered tools for artificial intelligence [138, 136].

Part II

Insights to Reveal Model Vulnerabilities

OVERVIEW

In the preceding part, our main focus was to decipher the inner workings of high-performing DNNs. Despite their efficacy, DNNs remain vulnerable, particularly to adversarial attacks. Perturbing inputs with minute, often indiscernible noise can mislead DNNs, resulting in incorrect outcomes. This vulnerability is particularly concerning in safety-critical applications, such as autonomous vehicles or healthcare systems. Thus, a critical question arises: How can we detect, interpret, and visualize these vulnerabilities in DNNs?

Part II introduces an interactive system that helps understand the susceptibility of DNNs to adversarial attacks, by visualizing how the attack permeates the model's internals (Chapter 5). This chapter is adapted from work that was published at VIS 2020 [89].

Chapter 5
BLUFF: Interactively Deciphering Adversarial Attacks on Deep Neural Networks. Nilaksh Das*, Haekyu Park*, Zijie J. Wang, Fred Hohman, Robert Firstman, Emily Rogers, Duen Horng Chau. (*Authors contributed equally). *IEEE Visualization Conference (VIS), 2020.*

Building on the insights gained from BLUFF, we expand our exploration into human-action recognition models, given their increasing prevalence in domains like home robotics, elder care, and security surveillance. We introduce another interactive system that explains and visualizes how attacks target human joints in human action recognition models, detailed in Chapter 6. This chapter is adapted from work that was published as an AAAI 2021 Demo [151].

Chapter 6
SKELETONVIS: Interactive Visualization for Understanding Adversarial Attacks on Human Action Recognition Models. Haekyu Park, Zijie J. Wang, Nilaksh Das, Anindya S. Paul, Pruthvi Perumalla, Zhiyan Zhou, Duen Horng Chau. *The AAAI Conference on Artificial Intelligence, Demo, 2021.*

CHAPTER 5

BLUFF: INTERACTIVELY DECIPHERING ADVERSARIAL ATTACKS ON DEEP NEURAL NETWORKS

Deep neural networks (DNNs) are now commonly used in many domains. However, they are vulnerable to *adversarial attacks*: carefully-crafted perturbations on data inputs that can fool a model into making incorrect predictions. Despite significant research on developing DNN attack and defense techniques, people still lack an understanding of how such attacks penetrate a model's internals. We present BLUFF, an interactive system for visualizing, characterizing, and deciphering adversarial attacks on vision-based neural networks. BLUFF allows people to flexibly visualize and compare the activation pathways for benign and attacked images, revealing mechanisms that adversarial attacks employ to inflict harm on a model. BLUFF is open-sourced and runs in modern web browsers.

5.1 Introduction

Deep neural networks (DNNs) serve as a major driving force behind recent technological breakthroughs [152, 153, 154, 155, 156, 157]. However, their susceptibility to *adversarial attacks* raises substantial concerns. Even minor, nearly imperceptible noise in inputs can mislead DNNs [18, 19, 20, 21], alarming in safety-critical applications like autonomous driving. Understanding these impacts on DNNs is vital [22, 23], but interpreting and ultimately defending against such attacks still remain fundamental research challenges [69]. Due to DNNs' complexity, pinpointing the components exploited by the attacks is challenging. Also, there is a lack of research in understanding how varying attack "strengths" influence neurons' activation patterns [158]. For example, it remains unknown if a stronger attack exploits the same neurons as a weaker attack, or if these sets are completely different.

To address the above challenges, we develop BLUFF (Fig 5.1), an interactive visualization tool for discovering and interpreting how adversarial attacks mislead DNNs into making incorrect decisions. Our main idea is to visualize **activation pathways** within a DNN traversed by the signals of benign and adversarial inputs. An *activation pathway* consists of neurons (also called *channels* or *features*) that are highly activated or changed by the input, and the connections among the neurons. BLUFF finds and visualizes where a model is exploited by an attack, and what impact the exploitation has on the final prediction, across multiple attack strengths.

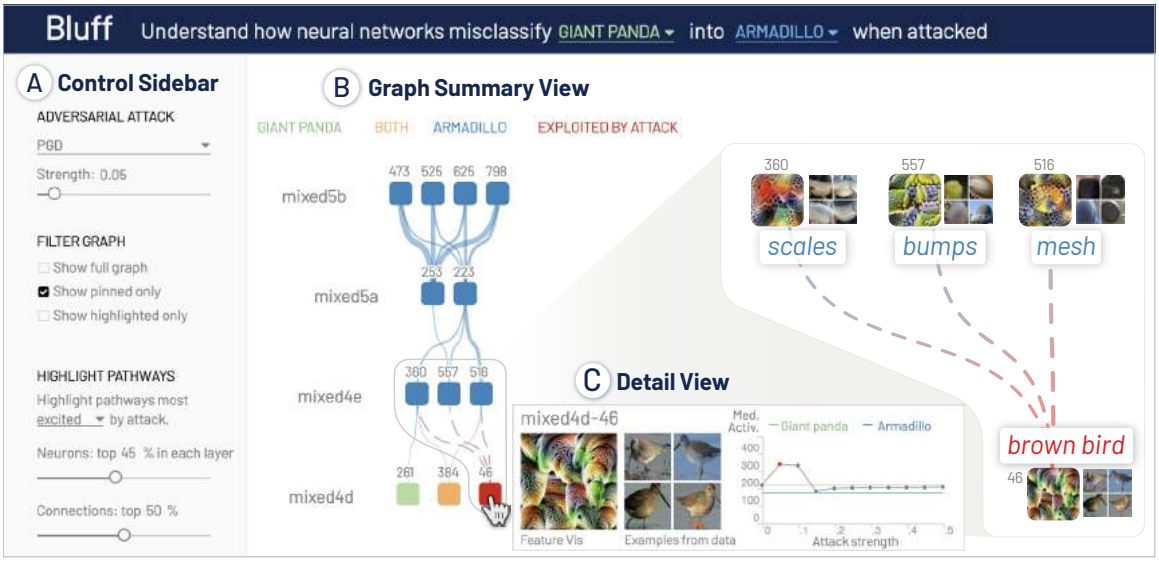


Figure 5.1: With BLUFF, users interactively visualize how adversarial attacks infiltrate a deep neural network, causing it to produce incorrect results. Here, a user can investigate the reasons why an InceptionV1 model misclassifies adversarial **giant panda** images, which have been manipulated using the *Projected Gradient Descent* (PGD) attack, as **armadillo**. PGD successfully distorts pixels to induce the "*brown bird*" feature, an appearance more commonly associated with armadillos (small, roundish, brown bodies) than pandas, activating additional concepts that contribute to the armadillo (mis)classification (e.g., "*scales*," "*bumps*," "*mesh*"). The *adversarial* pathways, formed by these neurons and their connections, overpower the benign panda pathways, ultimately leading to the incorrect classification. **(A) Control Sidebar** allows users to specify what data is to be included and highlighted. **(B) Graph Summary View** visualizes pathways most activated or changed by an attack as a network graph of neurons (each labeled by the channel ID in its layer) and their connections. When hovering over a neuron, **(C) Detail View** presents its feature visualization, representative dataset examples, and activation patterns over attack strengths.

We contribute:

- **BLUFF, an interactive system for summarizing and interpreting** how adversarial perturbations penetrate DNNs to induce incorrect outcomes in InceptionV1 [27], a large-scale prevalent image-classification model, over images from ImageNet ILSVRC 2012 [159]. To support reproducible research and broaden its access, we have open-sourced BLUFF at `https://poloclub.github.io/bluff`.
- **Visual characterization of activation pathway dynamics.** Adversarial perturbations manipulate activation pathways typically used for benign inputs to induce incorrect predictions. For example, an attack can *inhibit* neurons detecting important features for the benign class and *excite* those that exacerbate misclassification. BLUFF visualizes and highlights activation pathways exploited by an attack (Fig 5.2) and shows how they mutate and propagate through a network.

- **Interactive comparison of attack escalation.** BLUFF enables interactive comparison of activation pathways under increasing attack strengths, providing a new way for understanding the essence of an attack (e.g., common trends of an attack across all strengths) and its multi-faceted characteristics (e.g., various strategies that different strengths may employ).
- **Discovery usage scenarios.** We describe how BLUFF can help discover surprising insights into the vulnerability of DNNs, such as how unusual activation pathways may be exploited by attacks.

5.2 BLUFF: Deciphering Adversarial Attacks

5.2.1 Design Goals

Through a literature survey, we have identified the following four design goals (**G1**-**G4**) that guide BLUFF's development.

G1 Untangling activation pathways. *Benign* activation pathways can significantly overlap with *adversarial* pathways, as some neurons are "*polysemantic*," detecting multiple concepts at the same time [58, 60]. We aim to identify neurons that respond differently between benign and attacked inputs, to help discover where and how a model is exploited by an attack to induce incorrect predictions.

G2 Interpreting multiple activation pathways. Understanding the effects of adversarial attacks is core to developing robust defenses [19, 160, 158]. We aim to visualize high-level overviews of *benign* and *adversarial* activation pathways, and support drilling-down into subpaths, to help shed light on how specific groups of neurons are exploited to inflict harm on a model.

G3 Comparing attack characteristics. Existing works to interpret adversarial attacks on deep neural networks often focus on visualizing the activation patterns for a single adversarial input [97, 161]. We aim to visualize how the activation pathway changes as the attack strength varies, to help users gain deeper insight into how the attack works generally. Understanding model vulnerability under different attack strategies informs more robust defenses [130, 162, 163].

G4 Lowering barrier of entry for interpreting and deciphering adversarial attacks. The visualization community is contributing a variety of methods and tools to help people more easily interpret different kinds of DNNs [78, 56, 60, 97, 69, 164, 165]. Efforts that aim to support deciphering adversarial attacks, however, are relatively nascent [60, 69, 161]. We aim to make interpreting adversarial attacks more accessible to everyone, following the footsteps of prior success from the community.

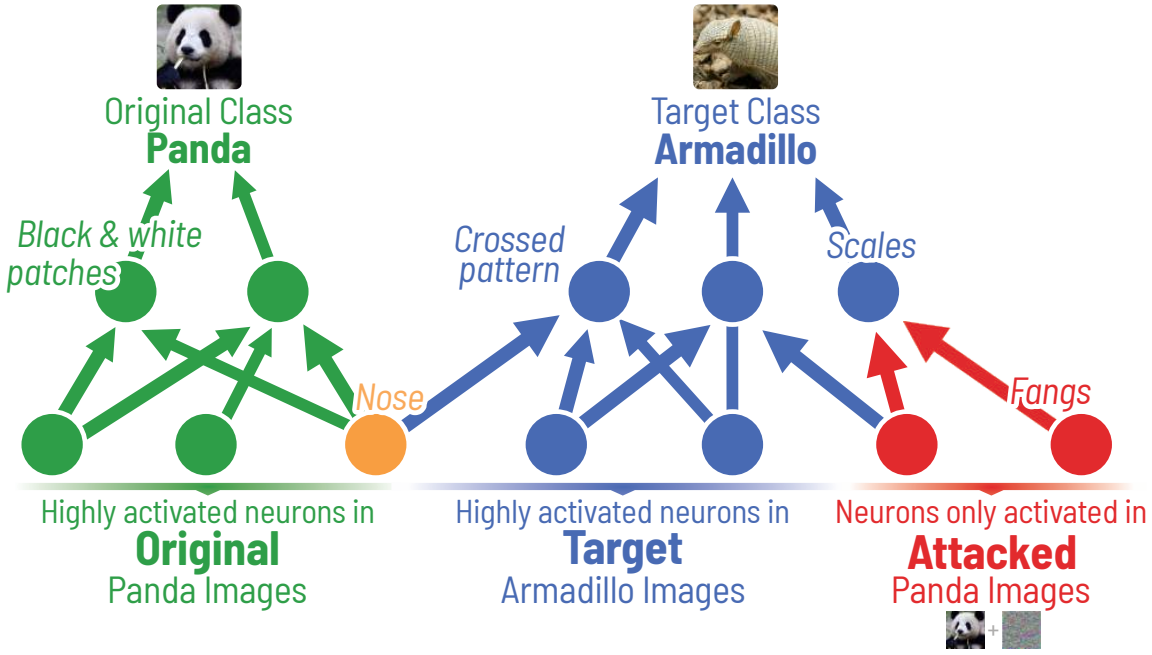


Figure 5.2: Adversarial attacks confuse DNNs to make incorrect predictions (e.g., misclassify benign *panda* as *armadillo*). BLUFF helps discover where such attacks occur and what features are used.

5.2.2 Background: Neuron Importance and Influence

To discover activation pathways triggered by benign and adversarial inputs, BLUFF identifies important neurons and influential connections among such neurons. Drawing inspiration from [78], BLUFF computes a neuron's *importance* based on how strongly it is activated by all inputs, and the *influence* between neurons based on the intensity of activation signals transmitted through the connection to the next layer. While providing a comprehensive interpretation of pathways within a DNN remains a challenge [67, 166, 167], recent studies [58, 78, 56] have shown that principal neuron activations in each layer effectively act as the basis vectors for the entire DNN's activation space. Thus, by characterizing *important* neurons' activations at every layer, we can achieve an indicative sampling of vital neurons throughout the network for a specific set of images.

Building upon this idea, BLUFF efficiently compiles activation pathways over *multiple* contexts, focusing on the most crucial neurons for: (1) benign images from the **original** class, on which the targeted attacks are performed; (2) benign images from the **target** class, which the attacks try to flip the label to; and (3) successfully **attacked** images for a particular attack strength (we support exploration with multiple strengths).

To begin, we consider the DNN model $\mathcal{M}$ (InceptionV1), where $Z^q \in \mathbb{R}^{H_q, W_q, D_q}$ is the output tensor of the q'th layer of $\mathcal{M}$. Here, H_q, W_q and D_q are the height, width and depth dimensions respectively. This implies that the layer has D_q neurons. We denote the d'th

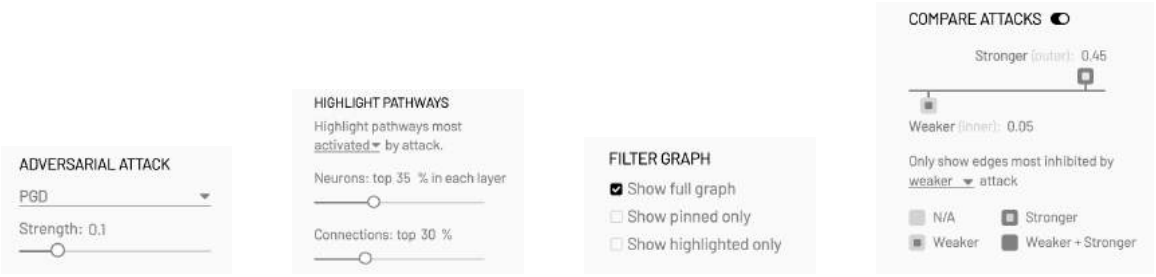
output channel (for d'th neuron) in the layer as $\mathcal{C}_q^d \in \mathbb{R}^{H_q, W_q}$. We index the values in the channel as $\mathcal{C}_q^d[h, w]$. Given an input image x_i, we find the maximum activation of each neuron induced by the image using the global max-pooling operation: $a_q^d[i] = \max_{h,w} \mathcal{C}_q^d[h, w]$. This represents the magnitude by which the d'th neuron in the q'th layer maximally detects the corresponding semantic feature from image x_i. This technique of extracting maximal activation as a proxy for semantic features has also demonstrated tremendous predictive power in the data programming domain [168]. Finally, we pass all images from each of **original**, **target** and **attacked** datasets. For each set, we aggregate $a_q^d[i]$ values for all images and quantify the importance of each neuron by the median value of such maximal activations. We use medians for summarizing the neuron importance, because they are less sensitive to extreme values. Consistent with the findings of [78], we observe that the maximal activation values are power law distributed, implying that only a small minority of neurons have highest importance scores. Hence to denoise inconsequential visual elements, for each layer, we empirically filter the 10 most important neurons for benign images of original and target classes, and 5 most important ones for attacked images of each attack strength (i.e., at most 50 important neurons across all 10 attack strengths).

To compute influence of a connection between two neurons, we measure the signal transmitted through the connection, computed by the convolution of the slice of the kernel tensor between the two neurons over the source neuron's channel activation. Since output from the ReLU activation function is used as the neuron activation in an InceptionV1 model, it implies that the neuron importance scores that are propagated are non-negative. Consequently, the model acts on these non-negative activation values. Hence, these activation values accumulate only for positively weighted convolution operations through consecutive layers, which has the effect of filtering out non-influential connections that may even originate from important neurons. Inspired by [78], we take the maximum value in the convolution for the influence of the connection. BLUFF deviates from [78] when aggregating influence values across several images. We characterize the connection by taking the median influence across all images from a given set. We take this approach since we want to summarize the influence characteristics across multiple datasets (**original**, **target** and **attacked** for different attack strengths), and each dataset is of a different size. Simple counting may skew the results towards a particular dataset while the median value provides a characteristic aggregation of the influence scores.

5.2.3 Realizing Design Goals in BLUFF's Interface

BLUFF's interface (Fig 5.1) consists of: **A. Control Sidebar** for selecting which data are included, filtered, highlighted, and compared; **B. Graph View** that summarizes and visualizes *activation pathways* as a graph; **C. Detail View** for interpreting the concept that a neuron has detected, via feature visualization, representative dataset examples, and activation patterns over attack strengths. In the header (Fig 5.1, top), users can select a pair of

original and **target** class. BLUFF then generates the main visualization in the Graph View for how neural networks misclassify **original** images as **target** images when attacked, by displaying the activation pathways of adversarial inputs.



(a) Attack Control Panel (b) Highlight Control Panel (c) Filter Control Panel (d) Comparison Control Panel

Figure 5.3: Control panels

Control Sidebar: Customizing Data Visualization Choices. Control Sidebar consists of four panels: Attack control panel, Highlight Control Panel, Filter Control Panel, and Comparison Control Panel (Fig 5.3).

Attack Control Panel allows users to manipulate the input images of the selected classes. Users can first select the adversarial attack method with a drop-down menu and then select the strength of the attack with a slider.

Highlight Control Panel allows users to select which type of activation pathways to visualize: pathways that are most *activated*, most *changed*, most *inhibited*, or most *excited* by an attack. Highlight Control Panel is equipped with two sliders for refined visualization. The first slider adjusts the percentage of neurons to be highlighted in each layer. As an illustration, by setting the slider to "35%" and opting for the "most activated pathways," BLUFF highlights the upper 35% of activated neurons across all categories: the original class, the target class, both classes, and those exploited by the attack. The second slider controls the percentage of connections that will be emphasized between layers. For example, by selecting "30%" along with the "most activated pathways" option, BLUFF highlights the top 30% of the most activated connections between neighboring layers, specifically from all positive connections among the highlighted neurons. The reason behind determining percentages layer-wise, rather than from all layers collectively, stems from the inherent variation in activation levels between layers. Taking the InceptionV1 model as an example, the lower layers exhibit neurons and connections that are substantially more active, making them not directly comparable to their activations in the deeper layers.

Filter Control Panel aims to enhance a more concentrated exploration of users navigating the visualization of activation pathways in BLUFF. Given that a dense collection of neurons and edges can become visually overwhelming, the Filter Control Panel mitigates such a visual clutter by offering three distinct filtering options: visualizing the full graph,

revealing only user-pinned neurons, or focusing exclusively on the highlighted neurons. When users pin a neuron, its identifying number is displayed above it for easy reference.

Comparison Control Panel enables users to compare two distinct attack strengths. When users switch to and activate the comparison mode, the range slider becomes available for setting these strengths: the left knob represents the milder attack, while the right knob corresponds to the stronger one. Additionally, users can decide which connections to visualize, whether they are influenced by the weaker or the stronger attack

Graph View: Visualizing Activation Pathway Summaries. Graph View (Fig 5.1B) summarizes InceptionV1's responses to inputs under different contexts in a unified graph. It focuses on InceptionV1's 9 mixed layers (mixed3a, mixed3b ... mixed5b), following existing interpretability literature [56, 100, 78]. Neurons important solely for the *original* class are presented as green nodes, those essential for the *target* class appear in blue, neurons significant for both classes are presented in orange, and those important for successfully attacked images are in red. Organized horizontally, the neurons are grouped by their roles in the sequence of original, both, target, and exploited by attack. Vertically, they align with the neural network layers: the uppermost row corresponds to the last mixed network layer (mixed5b), which is closest to the output layer, while the bottom aligns with the first mixed layer (mixed3a), near the input. Each neuron is visualized as a rounded rectangle and colored by its corresponding group. Each connection between two neurons is visualized as a curved line, where the width is linearly scaled by the influence values computed as described in Sec 5.2.2.

Out of all neurons, BLUFF highlights those that are most activated or changed based on the user-specified settings from the Highlight Control Panel. BLUFF highlights neurons by coloring them darker and connections by making others invisible (Fig 5.4). This highlighting approach naturally visualizes the pathways of activation signal passing through neurons and their connections, while preserving the context of all potentially relevant neurons nearby. This approach can also help compare the neurons that are highly activated by benign against adversarial images. All neurons displayed in BLUFF are already the most activated neurons in benign images, regardless of whether they are highlighted or not. On top of the neurons that are important for benign images, BLUFF highlights neurons that are highly activated by adversarial images. This helps distinguish neurons that are activated by benign as well as adversarial images, as compared to neurons only activated by benign images.

To help users more easily interpret the concepts that a neuron is detecting, alongside each neuron, BLUFF shows (1) a *feature visualization*, an algorithmically generated image that maximizes the neuron's activation, and (2) *dataset examples*, cropped from real images in the dataset, that also highly activate the neuron [56]. Hovering on a neuron shows the corresponding feature visualization and dataset examples as seen in Fig 5.1C, where adversarial images successfully induce the "*brown bird*" feature, an appearance more likely

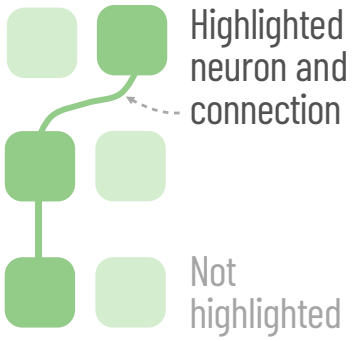


Figure 5.4: Visualization of highlighted neurons and connections

shared by an armadillo (small, roundish, brown body) than a panda, which in turn activates more features in subsequent layers that contribute to the (mis)classification of armadillo. These visual explanations help translate abstract activation pathways into the composition and flow of learned concepts (**G2**).

Detail View. When hovering on each neuron, a Detail View pops up to provide more information about the neuron (as seen in Figure 5.1C), display the feature visualization for the neuron and four example image patches to represent the detected concept the neuron detects. The Detail View also presents a line plot that displays the *median activation values* for: (1) images of the original class (green); (2) images of the target class (blue); and (3) adversarial images (gray). The horizontal axis represents the attack strength. Thus, each data point in the line plot represents a median activation value at a particular attack strength.

Visualizing Exploited Activation Pathways. An adversarial input is often a slightly perturbed version of a benign input, which means the activation pathways of an benign image and those of its adversarial counterpart would be similar at the input layer [69], yet decidedly different at the output layer — the *benign* pathways lead to the **original** prediction, while the *adversarial* pathways lead to the **target** prediction. Given the similar starting points but different outcomes, the adversarial activation pathways must have deviated from the benign pathways. BLUFF helps discover vulnerable neurons and connections that contribute to such deviations and the resulting misclassification, by highlighting the neurons and connections that are *excited* (or *inhibited*, oppositely) the most by an attack (Fig 5.1A) (**G1**). A pathway *excited* by an attack means its constituent neurons are activated more than expected (i.e., pathway contains more target features). Fig 5.1 shows an example of where the attack *excites* multiple features and connections to induce the target prediction of armadillo (e.g., "*scales*," "*bumps*," "*mesh*," and "*brown bird*" thanks to its similarity to armadillos' roundish, brown body). Computationally, in layer q, a neuron d's excitation amount is $\tilde{a}_d^q[attacked] - \tilde{a}_d^q[benign]$, where $\tilde{a}_d^q[benign]$ and $\tilde{a}_d^q[attacked]$ are the neuron's importance for some benign and attacked images respectively (as described in Sec 5.2.2).

Comparing Attacks with Varying Strengths. BLUFF offers the *Compare Attacks* mode that visualizes and compares the pathway differences between a weaker attack and a stronger

Figure 5.5: Visual encoding of neurons based on which attack strengths they respond to

attack (**G3**). BLUFF visually encodes the neurons based on which attack strengths they responded to (Fig 5.5), drawing inspiration from Alper et al [169]. Each neuron consists of an inner and an outer rectangle: the *inner* rectangle is colored when the neuron is in the activation pathways of the *weaker* attack; whereas the *outer* rectangle is colored when the neuron is in the activation pathways of *stronger* attack. Thus, our design can visually encode all four possible comparison results, enabling us to use hue to represent the four neuron contexts (i.e., neurons targeted by either weaker or stronger attacks, or those targeted by both or none of them). Our terminology for *weaker* and *stronger* attacks are relative, as we do not assert any explicit threshold for weak or strong attack strengths.

Cross-platform deployment through standard web technologies. To facilitate reproducible research and enhance accessibility, BLUFF uses standard web technologies including HTML, CSS, JavsScript, and D3.js. BLUFF can be accessed from any modern web browser (**G4**) at `https://poloclub.github.io/bluff`. We ran all the computations for neurons' importance and connections' influence on a NVIDIA DGX-1 workstation equipped with 8 GPUs (each with 32GB memory), 80 CPU cores, and 504GB RAM.

5.3 Discovery Usage Scenarios

In the forthcoming demonstrations, we present how BLUFF enhance the understanding of adversarial attacks while also revealing the strategies these attacks employ to confuse a DNN. Our chosen scenarios draw upon the vast pool of 1000 classes available in the ImageNet dataset [159], which comprises approximately 1.2 million images.

5.3.1 Understanding How Attacks Penetrate DNNs

Consider a DNN classifier tasked with identifying snakes, such as the highly venomous *diamondback snake* and the milder *vine snake* whose venom results in only slight swelling. In Fig 5.6, BLUFF's Graph View uncovers the unusual tactic used by adversarial diamondback images: they activate unexpected neural pathways, leading to an incorrect prediction

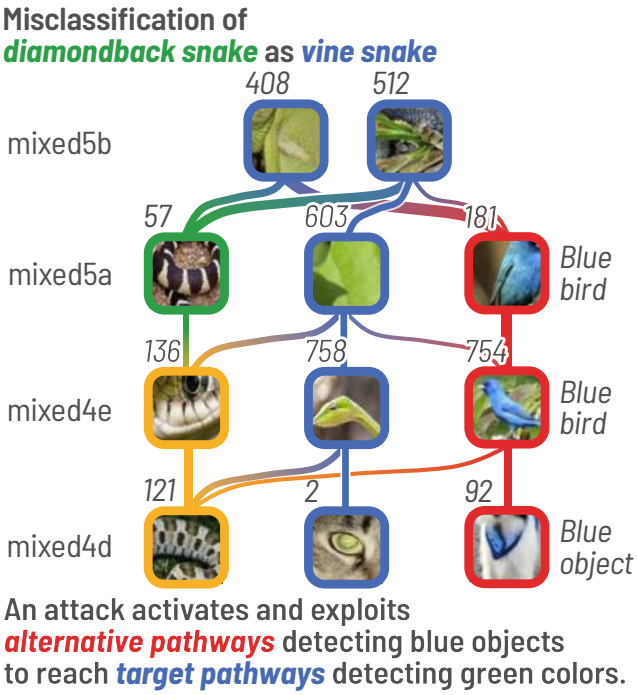


Figure 5.6: BLUFF helps users understand how an attack penetrates a model, by visualizing activation pathways that are additionally exploited by the attack. In this example, BLUFF highlights the neurons and connections that PGD attack exploits (red) to make the model confuse adversarial **diamondback snake** images as **vine snake**.

of vine snakes. Intriguingly, these activated or exploited neurons are tuned to detect the *"blue color"* (e.g., blue birds as seen in Fig 5.6, right column).

This is surprising because vine snakes, which is the intended target class of the attack, are primarily green, not blue. This finding implies that the PGD attack may be taking advantage of the *"blue color"* concept pathway as an alternative route to connect with the pathways for vine snake characteristics like *"green leaves"* and *"green bumps"* (Fig 5.6, middle column). We also noticed that PGD uses *"snake-like"* pathways that are important for both source and target classes (Fig 5.6, left column), which is reasonable given that both the original and target classes are snakes. Identifying these neural pathways that are taken advantage of during an attack offers fundamental insights for devising robust defense mechanisms, potentially by neutralizing these alternative neural routes. Another illustrative example can be found in Fig 5.1, where pathways associated with the *adversarial armadillo* class overshadow those of the *benign panda* class, resulting in a misclassification.

5.3.2 Variation in Attack Strategies Based on Intensity

Adversarial attacks vary in intensity from subtle to pronounced disturbances, as illustrated in Fig 5.7. Does the strategy behind an attack change according to its intensity, or does it remain consistent?

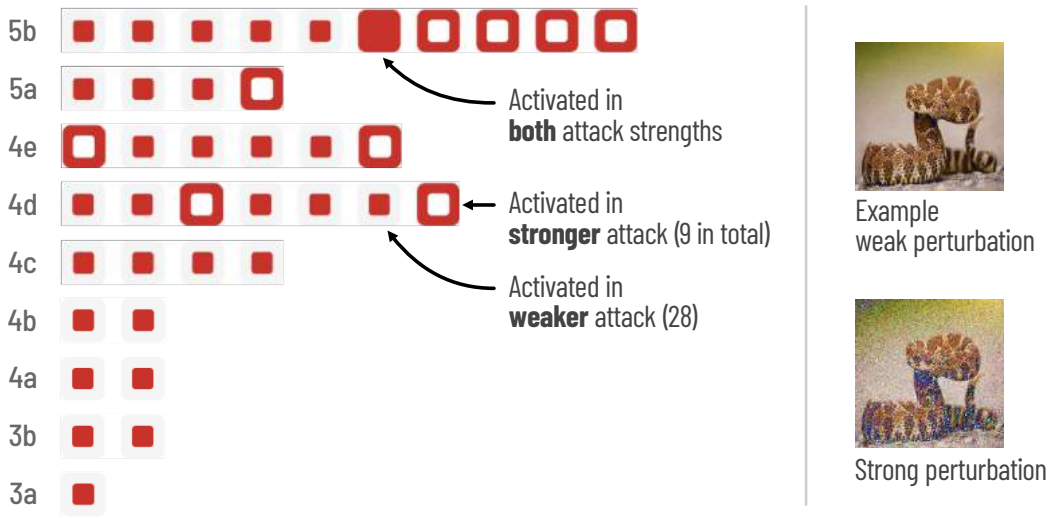


Figure 5.7: Using BLUFF's "Compare Attacks" mode, we examine PGD's different strategies for misclassifying *diamondback* images into *vine snake* class for two attack strengths (0.1 vs 0.5). The weaker attack exploits more alternative neurons (i.e., features that are not typically activated by benign inputs) than the stronger attack does.

With BLUFF's *Compare Attacks* mode, users can conduct a comparative analysis of these variations. To exemplify, let's evaluate attacks on *diamondback* images with the aim of misclassifying them as *vine snake*. By adjusting the attack strength to the lower end (0.1) and the higher end (0.5), and focusing on the top 30% of the most activated neurons in each layer, an intriguing pattern emerges in Fig 5.7. Specifically, weaker attacks appear to exploit a substantial number of red neurons (28 in total). This contrasts starkly with the stronger attack, which only affects 9 of such neurons. These neurons are not particularly crucial to identifying either snake variety but are markedly influenced by the attack. Diving deeper into the example image patches of these neurons, we noticed that the images encompass a diverse set of semantic themes, such as textitspider legs, *blue bird* and *car hood*, seemingly unrelated to snakes.

Moreover, this pattern was not exclusive to the snake images. When *ambulance* were attacked to be misclassified as *street sign*, the weaker exploited 42 red neurons compared to the 16 exploited by its stronger counterpart. Similarly, in the case of attacks turning *panda* into *armadillo*, 29 neurons were exploited by the weaker attack, and only 8 were exploited by the stronger attack.

Piecing together these observations, we infer that milder attacks tend to utilize a broader array of unrelated semantic concepts to trigger misclassifications. This phenomenon can be metaphorically likened to a strategy of "death by a thousand cuts," where numerous slight disturbances on neurons collectively achieve the misclassification.

5.3.3 Correlating Class Similarity with Exploitation Intensity

An adversarial attack changes a data instance's predictions from its original class to a target class of the attacker's choosing. However, some class changes may be "easier" (e.g., from one type of dog to another), and some "harder" (e.g., from an animal to a vehicle). How does class similarity correlate with the magnitude of changes that an attack needs to induce inside a model?

Dissimilar class pairs → strong neuron inhibition. While attacks generally inhibit features from the original class, so that features of the target adversarial class are relatively detected more, we found that the magnitude of neuron inhibition is much higher when the original and target classes are very different. For example, Fig 5.8-left shows that adversarial *ambulance* images need to *strongly* inhibit car-related neurons (i.e., big drop in their activation) to misclassify such images as *street signs* due to strong class dissimilarities. Conversely, as shown in Fig 5.8-right, adversarial *brown bear* images only need to *mildly* inhibit brown-fur neurons to induce misclassification of black bear due to close class resemblance. We made similar observations for other dissimilar (e.g., *giant panda* vs. *armadillo*), and similar class pairs (e.g., *vine snake* vs. *green snake*).

Dissimilar classes → more neurons exploited. BLUFF helps reveal *unusual* pathways that an attack exploits to induce incorrect predictions. Such unusual pathways (colored red in Graph View) typically consist of neurons and connections that are *neither* commonly traversed by *benign original* images, nor *benign target* images; rather they are exploited unexpectedly by the adversarial images. We found that these unusual pathways are more heavily used — and they consist of more neurons and connections — when the original and target classes are dissimilar (e.g., *ambulance* vs. *street sign*; *giant panda* vs. *armadillo*). The unusual pathway for the *ambulance* class pair contains 136 neurons, and the *giant panda* pair contains 85, across all attack strengths. For similar class pairs (e.g., *brown bear* vs. *black bear*; *vine snake* vs. *green snake*, both with green bodies), their unusual pathways are dramatically more compact, 28 neurons for the *bear*s, and 16 for the *snake* for all attack strengths.

5.4 Conclusion and Future Work

We present BLUFF, an interactive system for visualizing, characterizing, and deciphering adversarial attacks on DNNs. We believe our visualization, summarization, and comparison approaches will help promote user understanding of adversarial attacks, and support discoveries to design a proper defense. Our next step is to use BLUFF to help construct robust defenses against attacks. We plan to extend BLUFF to support *interactive neuron editing* (e.g., "deleting" a neuron from model), so that the user may empirically identify and act on vulnerable neurons and observe the effects on the resulting pathway and prediction in

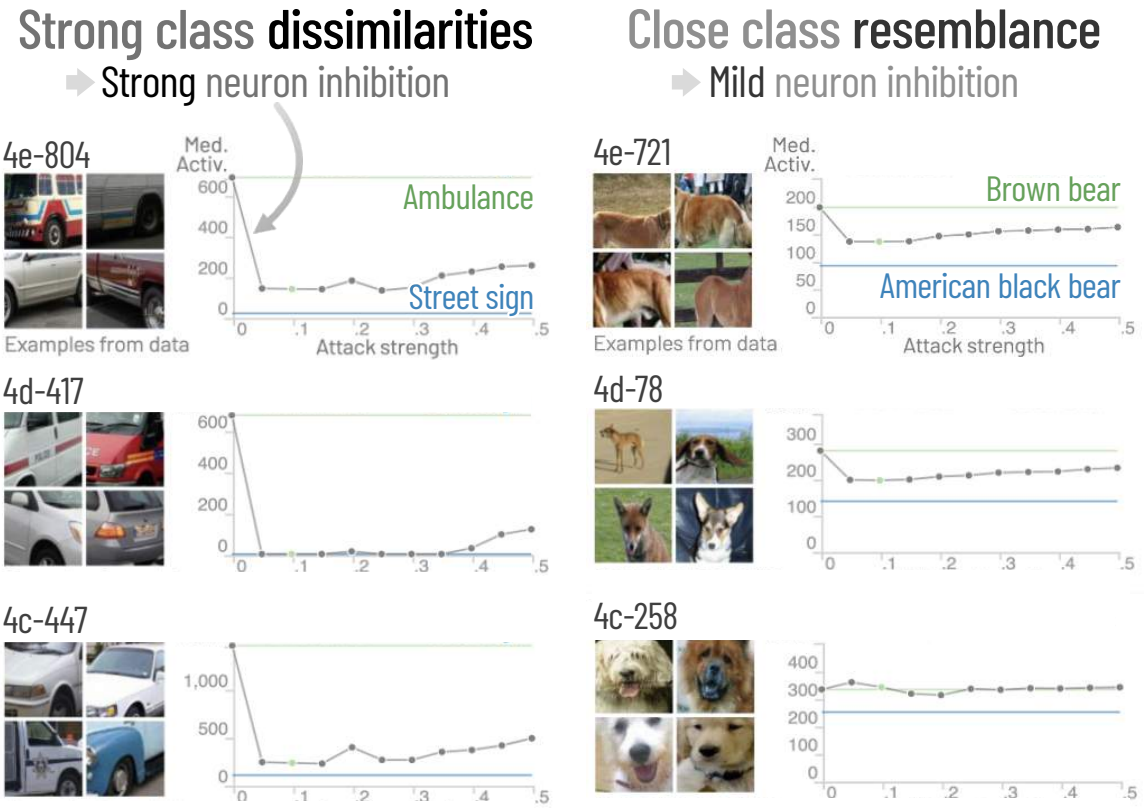


Figure 5.8: The most inhibited neurons for the *dissimilar* class pair (ambulance, street sign), and a *similar* class pair (brown bear, black bear). **Left:** adversarial *ambulance* images need to *strongly* inhibit car-related neurons (i.e., big drop in their activation) to misclassify such images as *street signs* due to strong class dissimilarities. **Right:** adversarial *brown bear* images only need to *mildly* inhibit brown-fur neurons to induce misclassification of *black bear* due to close resemblances.

real-time. We also plan to extend BLUFF to work for adversarially-trained models [19, 160, 158], to help people gain deeper insights that explain their robustness. Additionally, after receiving positive preliminary feedback from researchers, students and collaborators who were given the opportunity to try out BLUFF, we plan to conduct user studies to evaluate our tool's usability and functionality.

CHAPTER 6

SKELETONVIS: INTERACTIVE VISUALIZATION FOR UNDERSTANDING ADVERSARIAL ATTACKS ON HUMAN ACTION RECOGNITION MODELS

Skeleton-based human action recognition is becoming pivotal for video applications, spanning domains such as home robotics, eldercare healthcare systems, and surveillance. Yet, the susceptibility of these models to adversarial attacks raises red flags, especially when considering their integration into safety-critical environments. Gaining insight into the mechanisms by which these attacks can deceive human action recognition models is vital for devising robust defenses. In response, we present SKELETONVIS, the first interactive system that visualizes the intricacies of such adversarial tactics, fostering a deeper understanding of their operation.

6.1 Introduction

Skeleton-based human action recognition is increasingly vital in video-centric applications, ranging from home robotics to eldercare and surveillance, as evidenced by several studies [1, 2, 170]. These models, which analyze the complex movements and interconnections of human joints, demonstrate high predictive accuracy [1]. However, they face a significant challenge that they are vulnerable to adversarial attacks; subtle perturbations, almost imperceptible to the human eye, can trick these models into making incorrect predictions [171, 172]. This vulnerability issue is of particular concern in safety-critical applications like autonomous driving systems.

To enhance the resilience of these models against such attacks, understanding their susceptibility is crucial. Leveraging the power of interactive visual analytics systems, which have proven effective in exposing deep learning model vulnerabilities [89, 90, 173], we introduce SKELETONVIS (see Fig 6.1). SKELETONVIS is the first interactive visualization tool designed specifically to explain how adversarial attacks manipulate inputs to cause incorrect model predictions.

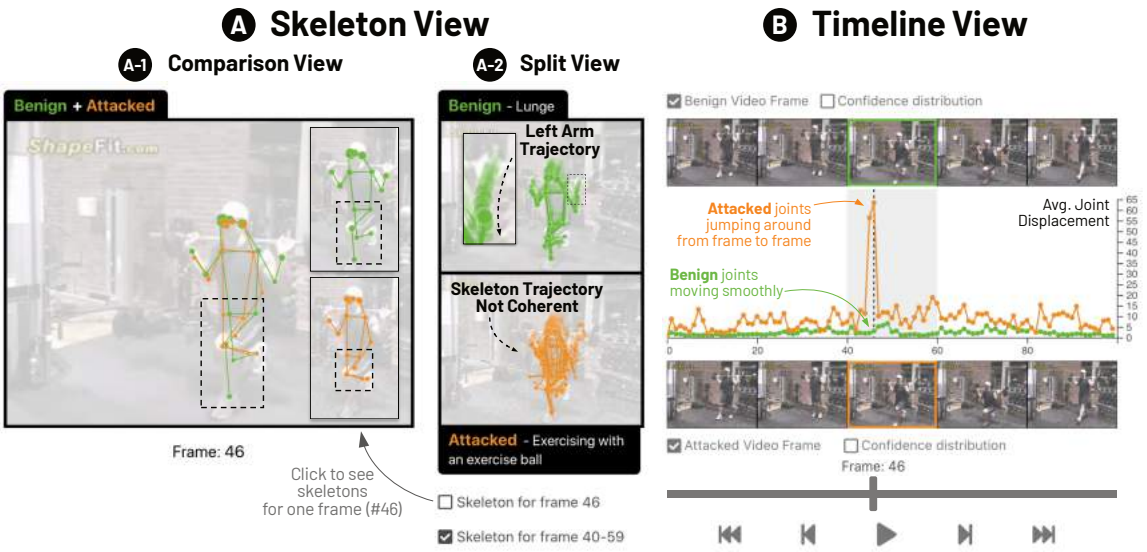


Figure 6.1: The interface of SKELETONVIS, visualizing how the *Fast Gradient Method* manipulates the left foot joints detected by the Detectron2 Keypoint R-CNN model. (A) The *Skeleton View* shows the joints perturbed to unexpected locations. (B) *Timeline View* reveals the attacked joints spuriously jumping around from one frame to the next, leading to a "spike" in the average joint displacement across attacked frames. These manipulations finally sway the ST-GCN action detection model into misclassifying the attacked frames as "exercising with exercise ball," instead of the correct "lunge" classification.

6.2 The SKELETONVIS System

SKELETONVIS is developed to provide in-depth insights into adversarial attacks targeting skeleton-based human action recognition systems. It operates by analyzing two types of inputs: a benign video clip, such as one showing a *person lunging*, and its "attacked" counterpart, which might depict a seemingly similar activity like *exercising with a gym ball*. These attacked versions are meticulously crafted by applying subtle pixel-level perturbations to the original, benign clips (as depicted in Fig 6.1A).

SKELETONVIS visualizes the human joints detected in both benign and adversarial videos, highlighting how these joints may become misaligned, thus leading to incorrect predictions. SKELETONVIS also measures and displays atypical patterns in the manipulated frames, enabling users to pinpoint the specific frames that are targeted by attacks. Furthermore, SKELETONVIS integrates quantitative measurements to highlight irregularities in the adversarial frames, thereby guiding users to the specific frames most impacted by the attack, as illustrated in Fig 6.1B.

6.2.1 Extracting Human Joints to Predict Action and Visualize Attack

SKELETONVIS offers a visualization interface to interpret the impacts of adversarial attacks on an end-to-end skeleton-based human action recognition system. The human-action recognition system operates in two fundamental phases. First, it identifies 17 pivotal human joints in each video frame, using Detectron2's body joint R-CNN model [174], which produces a spatial confidence map for every identified joint. The extracted joint information forms the foundational visualization layer within SKELETONVIS, allowing users to observe and analyze the spatial distribution of detected joints in each frame. Second, after joint extraction, it infers the human action by integrating the spatial confidence map of the detected joints, by employing ST-GCN [1] which is a state-of-the-art skeleton-based action detection model. As adversarial perturbations are introduced to this end-to-end system, SKELETONVIS visualizes their effects, specifically highlighting discrepancies and anomalies in joint detections and their subsequent influence on action inferences.

6.2.2 *Skeleton View*: Explaining How Attacks Manipulate Detection of Human Joints

Skeleton View, as depicted in Fig 6.1A, visually explains how the attacks are manipulating the human joint positions to cause misclassification. This explanation is facilitated through two distinct visualization models: the *Comparison View* (Fig 6.1A-1) that overlays the benign and adversarial joint positions for direct comparison and the *Split View* (Fig 6.1A-2) that presents these positions separately for clearer distinction.

Comparison View enables a *spatial* comparison between benign and adversarial frames, by superimposing the two sets of detected joints. For example, by revealing that the left foot is detected in unexpected locations in adversarial frames, users can more easily glean insights into why the pose detection model misclassifies the video (e.g., person lunging) as an incorrect action (e.g., exercising with an exercise ball).

Split View compares *movement* of the human joints in the benign and adversarial clips. For a clip, we visualize the trajectory of a human joint as follows. Let $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T$ be the sequence of the coordinates of a human joint in a clip. We display the movement as in Fig 6.1A-2, where joints from earlier frames (i.e., $\mathbf{x}_1, \mathbf{x}_2, ...$) are visualized as more transparent dots, and those from frames (i.e., $\mathbf{x}_T, \mathbf{x}_{T-1}, ...$) are visualized as more opaque dots. The joint trajectories and frames of the benign video are shown at the top, and those of the adversarial videos are shown at the bottom.

6.2.3 *Timeline View*: Visualizing Abnormal Signals from Adversarial Attacks

To help users discover attacked frames, SKELETONVIS quantifies and visualizes the pose detection models' responses to benign and attacked videos in the *Timeline View* (Fig 6.1B), enabling users to interactively compare and more easily pinpoint frames that they need to be wary of.

At the upper section of *Timeline View*, users can closely inspect a *video segment* (defined as a specific range of frames) by selecting the segment's thumbnail. This action not only highlights the chosen thumbnail but also displays its associated average joint displacement values in a line chart. In this chart, the horizontal axis denotes the sequence of frames, while the vertical axis quantifies the displacement occurring from one frame to the next.

At the bottom of *Timeline View*, users can interact with controls which are similar to those found in standard video players, to select specific frames for analysis. The interactive timeline consists of a *frame slider* and *time buttons*. Moving the knob on the *frame slider* sets the selected frame in *Comparison View* (Fig 6.1A-1) and the segment in *Split View* (Fig 6.1A-2). Pressing the "Play" button pauses or resumes the video playback. The "Fast backward" and "Fast forward" buttons at both ends bring users to the very beginning and the very end of the clip respectively. The "Forward" and "Backward" buttons step forward or backward by one frame.

6.3 Conclusion

SKELETONVIS is the first-of-its-kind interactive system that visualizes how an adversarial attack affects a human action recognition system with videos. By employing a dual visualization strategy with *Skeleton View* and *Timeline View*, it visualizes and quantifies spatial and temporal discrepancies in joint detections and their subsequent influence on human action inferences. Such a comprehensive visualization becomes helpful for understanding and potentially mitigating the impact of adversarial perturbations on skeleton-based human action recognition systems.

Part III

Scalable Discovery of Concept Evolution During Training

OVERVIEW

The previous two parts focus on the interpretation of fully developed DNNs, examining their learned concepts and potential vulnerabilities. Yet, most research, including these prior parts, primarily focuses on insights after the training has concluded [9, 10].

The post-training approaches offer limited insights into how DNNs evolve as they are trained, even though interpreting the model evolution can be crucial for efficient network training monitoring [8, 9]. There is a discernible knowledge gap regarding the models' developmental progress during training and how it ties to issues such as limited generalizability [11, 12, 13] or convergence failures [14, 15]. Relying solely on post-training interpretations can impede real-time timely interventions and resource wastage, especially if the training fails to realize the intended outcomes [16, 17]. How can we help people interpret the dynamic evolution of a DNN as it is trained?

In Part III, we present the first-of-its-kind general unified interpretation framework of DNNs that reveals the inception and evolution of learned concepts during training (Chapter 7). This chapter is adapted from work that was published at CIKM 2023 [175].

Chapter 7
Concept Evolution in Deep Learning Training: A Unified Interpretation Framework and Discoveries. Haekyu Park, Seongmin Lee, Benjamin Hoover, Austin Wright, Omar Shaikh, Rahul Duggal, Nilaksh Das, Kevin Li, Judy Hoffman, Duen Horng Chau. *International Conference on Information and Knowledge Management, 2023*

CHAPTER 7

CONCEPT EVOLUTION IN DEEP LEARNING TRAINING: A UNIFIED INTERPRETATION FRAMEWORK AND DISCOVERIES

We present CONCEPTEVO, a unified interpretation framework for deep neural networks (DNNs) that reveals the inception and evolution of learned concepts during training. Our work addresses a critical gap in DNN interpretation research, as existing methods primarily focus on post-training interpretation. CONCEPTEVO introduces two novel technical contributions: (1) an algorithm that generates a unified semantic space, enabling side-by-side comparison of different models during training, and (2) an algorithm that discovers and quantifies important concept evolutions for class predictions. Through a large-scale human evaluation and quantitative experiments, we demonstrate that CONCEPTEVO successfully identifies concept evolutions across different models, which are not only comprehensible to humans but also crucial for class predictions. CONCEPTEVO is applicable to both modern DNN architectures, such as ConvNeXt, and classic DNNs, such as VGGs and InceptionV3.

7.1 Introduction

Interpreting the inner workings of Deep Neural Networks (DNNs) has become crucial for instilling trust in the models [26], debugging them [176], and guarding against harms such as embedded bias or adversarial attacks [89, 177, 178]. As a fundamental type of DNN, convolutional neural networks have garnered significant interest in understanding their internal mechanism. Saliency-based interpretation methods aim to identify important image regions for predictions [45, 42]. Concept-based interpretation methods identify ***concepts*** detected by DNNs, such as "*dog face*" in Fig 7.1, and their role in forming higher-level concepts and predictions [80, 58, 62, 47, 60]. These methods connect a concept with sets of images or image patches that explain the concept, using shared visual characteristics among the images to enhance human understanding of the concept [179, 56, 62]. Neuron-level concept interpretation methods focus on concepts that elicit strong activation in that neuron [56, 179, 80].

However, existing approaches predominantly focus on post-training interpretation [9, 10], providing limited insights into their training processes. Crucially, understanding the progression of concepts detected by individual neurons, which we refer to as the neuron's **concept evolution**, and its association with model deficiencies like poor generalizability

Figure 7.1: CONCEPTEVO creates a unified semantic space that enables side-by-side comparison of different models during training (top: VGG19; middle: InceptionV3; bottom: ConvNeXt). CONCEPTEVO embeds and aligns neurons (dots) that detect similar concepts (e.g., dog face, circle, car wheel) to similar locations.

[11, 12, 13] or convergence failures [14, 15] remains underexplored. Relying solely on post-training interpretation poses challenges for real-time discovery and diagnosis during training, potentially wasting time and resources [16, 17], if the training ultimately fails to achieve desired outcomes. Interpreting the DNN training process also enhances effective monitoring [77, 8, 180, 181].

To fill these gaps, our work contributes as follows:

1. **CONCEPTEVO, a unified interpretation framework that reveals the inception and evolution of concepts during DNN training** (Sec 7.2), with two novel technical contributions[1]:
   - An algorithm that generates a unified semantic space that enables side-by-side com-

[1]CONCEPTEVO has been made open source: https://github.com/poloclub/ConceptEvo.

parison of different models during training (Fig 7.1, 7.2). CONCEPTEVO is applicable to both modern ConvNeXt and classic DNNs like VGGs and InceptionV3.
- An algorithm that discovers and quantifies important concept evolutions for class predictions (Fig 7.3).

2. **Extensive evaluation** (Sec 7.3). A large-scale human-based experiments involving 260 participants and quantitative experiments demonstrate that CONCEPTEVO identifies concept evolutions that are not only meaningful to humans but also important for class predictions.
3. **Discoveries on model evolution** (Sec 7.3.5). We highlight how CONCEPTEVO aids in uncovering potential issues during model training and provides insights into their causes, such as: (1) severely harmed concept diversity caused by incompatible hyperparameters (e.g., overly high learning rate) as shown in Fig 7.2b; and (2) slowly evolving concepts despite rapid increases in training accuracy in overfitted model as shown in Fig 7.2c.

7.2 Method

7.2.1 Desiderata of Interpreting Concept Evolution

D1 General interpretation of concept evolution across different models. Comparing the training of different models is essential for determining which model is trained better or which training strategy is more effective [11, 182]. Thus, we aim to develop a general method that enables side-by-side comparison and interpretation of concept evolution across different models. (Sec 7.2.2)

D2 Revealing and quantifying important evolution of concepts. We aim to identify internal changes that significantly impact the prediction of a specific class, as understanding the most influential components can lead to effective model improvements [183]. For example, we seek to determine the importance of a neuron's concept evolution, such as the transition from "brown color" to "brown furry leg" in the prediction of a "brown bear" class. We aim to automatically discover these important changes in concepts for class predictions. (Sec 7.2.3)

D3 Discoveries. Can the interpretation of how a model evolves help identify training problems and provide insights for addressing them, advancing prior work that focuses on interpreting and fixing models post-training [183]? For example, can we help determine if a model's training is on the right track and if interventions are necessary to improve accuracy? (Sec 7.3.5)

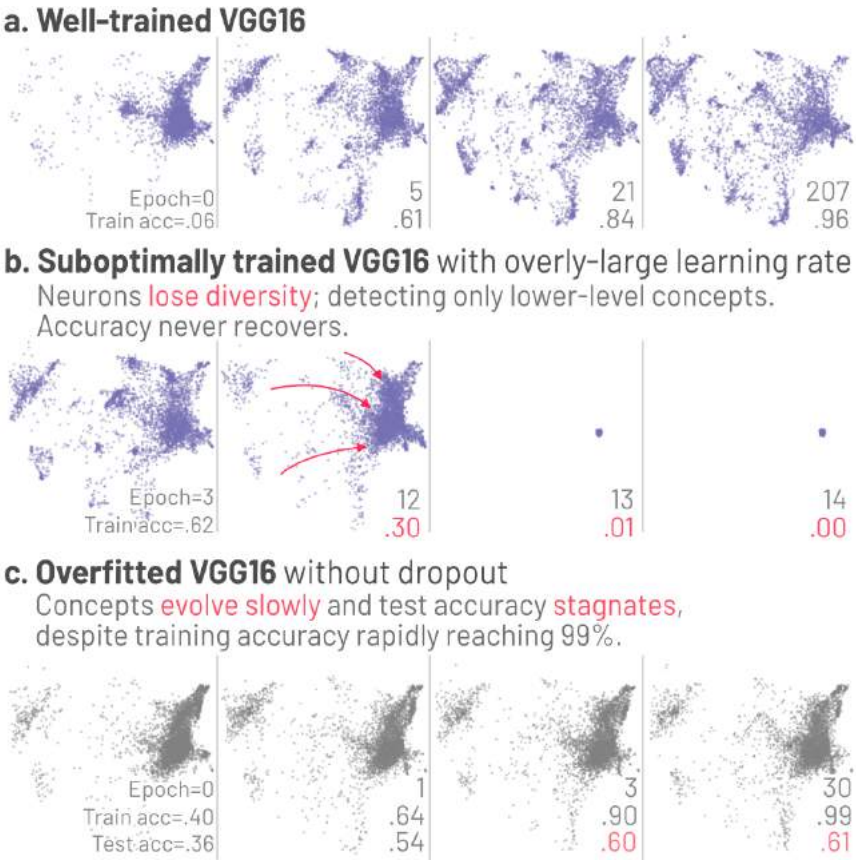


Figure 7.2: CONCEPTEVO identifies potential training issues. (a) A well-trained VGG16 shows gradual concept formations and refinements. (b) A VGG16 suboptimally trained with a large learning rate, rapidly losing the ability to detect most concepts. (c) An overfitted VGG16 without dropout layers, showing slow concept evolutions despite rapid training accuracy increases. We abbreviate "top-5 training/test accuracies" as "train/test acc."

7.2.2 General Interpretation of Concept Evolution

We desire an interpretation of model evolution that is comparable across different models. However, direct comparison between concepts in different models at different training stages is challenging. Different models are independently trained; thus, the learned concepts are not aligned by default. Even for the same model, activation patterns can change considerably over training epochs.

To address this challenge, we propose a two-step method. In step 1, we create a base semantic space that captures the concepts identified by a *base model* at a specific training epoch. This semantic space serves as a fundamental reference for concept representation. In step 2, we project the concepts from other models spanning all epochs onto the base semantic space, resulting in a **unified semantic space** where similar concepts across different models and epochs are mapped to similar locations.

We selected an optimally, fully trained model as our base model to ensure comprehen-

sive concept coverage. For example, we used a fully trained VGG19 model [184] as the base model for Fig 7.1 and 7.2.

Step 1: Creating the base semantic space. To create the *base semantic space*, we use neurons as a unit to identify and represent concepts, inspired by studies that demonstrate neurons' selective activation for specific concepts [183, 56, 99]. By using neurons, we can pinpoint areas of interest in models, enabling focused troubleshooting, particularly in identifying abnormal training patterns within specific groups of neurons. Building on prior work [80], we embed neurons that strongly respond to common inputs in similar locations. As neuron-concept relationships may not always be one-to-one [58, 185], we aim to generalize to many-to-many relationships. For example, polysemantic neurons responsive to multiple concepts are embedded between those concepts.

Step 1.1: Finding stimuli. CONCEPTEVO creates stimuli for each neuron by collecting a set of k images that result in the highest maximum in the neuron's activation map. For neurons associated with a single concept, their stimuli will be more alike, while for polysemantic neurons, their stimuli may consist of multiple concepts.

Step 1.2: Sampling frequently co-activated neuron pairs. CONCEPTEVO creates a multiset D, which consists of sampled pairs of strongly co-activated neurons from the base model M_b at epoch t_b. First, for each image $\mathbf{x}$, it creates a list of neurons that are strongly co-activated by $\mathbf{x}$, by collecting neurons with $\mathbf{x}$ in their stimuli. Next, it randomly shuffles each list of co-activated neurons and samples neuron pairs using a sliding window of length two over the shuffled neurons. The sampled neuron pairs are added to D. This sampling process is repeated E times to obtain diverse neuron pairs. Note that a specific neuron pair can appear multiple times in D, with their frequency of appearance increasing as more images are shared by their stimuli. This leads to a closer embedding of more frequently co-activated neurons in the unified semantic space.

Step 1.3: Learning neuron embedding. The objective function, defined by Eq (7.1), represents a negative log likelihood to learn neuron embeddings; intuitively, (1) co-activated neuron pairs with a larger inner product (and spatially closer embeddings) are more likely to indicate similar concepts, while (2) randomly paired neurons with a lower inner product (and spatially farther embeddings) are less likely to be conceptually similar. The randomly paired neurons serve as negative examples, enabling high-quality vector representations of concepts, similar to the negative sampling approach used in Word2Vec algorithm [110, 186]. This neuron embedding approach allows for the representation of many-to-many relationships between neurons and concepts. For example, a polysemantic neuron, which is co-activated by multiple distinct groups of neurons representing different concepts, is attracted towards these groups, resulting in its spatial location between them. In the objective function, $\mathbf{v}^t_{n,M}$ is an embedding of neuron n in model M at epoch t. r is a randomly selected neuron. R is the number of randomly sampled neurons for each co-activated neuron

pair in D. $\sigma(\cdot)$ is the sigmoid function (i.e., $\sigma(x) = 1/(1+e^{-x})$).

$$J_1 = -\sum_{(n,m)\in D} \Bigg(\log\left(\sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{m,M_b})\right) + \sum_{r=1}^{R} \log\left(1-\sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\right) + \sum_{r=1}^{R} \log\left(1-\sigma(\mathbf{v}^{t_b}_{m,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\right) \Bigg) \tag{7.1}$$

We randomly initialize the neuron embeddings and learn the embeddings by gradient descent. Eq (7.2) and (7.3) present the derivative to update the neuron embeddings. Further details about neuron embedding algorithm are described in Appendix A.

$$\frac{\partial J_1}{\partial \mathbf{v}^{t_b}_{n,M_b}} = (1-\sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{m,M_b}))\mathbf{v}^{t_b}_{m,M_b} - \sum_{r=1}^{R} \sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\ \mathbf{v}^{t_b}_{r,M_b} \tag{7.2}$$

$$\frac{\partial J_1}{\partial \mathbf{v}^{t_b}_{m,M_b}} = (1-\sigma(\mathbf{v}^{t_b}_{n,M_b} \cdot \mathbf{v}^{t_b}_{m,M_b}))\ \mathbf{v}^{t_b}_{n,M_b} - \sum_{r=1}^{R} \sigma(\mathbf{v}^{t_b}_{m,M_b} \cdot \mathbf{v}^{t_b}_{r,M_b})\ \mathbf{v}^{t_b}_{r,M_b} \tag{7.3}$$

Step 2: Unifying the semantic space of different models at different epochs.

Step 2.1: Image embedding. Different models, with varied architectures and neurons, can share the commonality of being trained on the same dataset. Leveraging this, we consider that if two neurons from different models are strongly activated by the same inputs, they likely detect the same concept. To represent neurons' concepts across models, we use image embeddings as a bridge: we compute image embeddings that approximate the original neuron embeddings in the base model, and these image embeddings are then used to approximate the neuron embeddings in other models.

A neuron's embedding typically represents a more detailed concept (e.g., car wheel as shown in Fig 7.1) extracted from the entire images (e.g., car images) that include various concepts (e.g., car wheels, loads, and more). Thus, we consider that collective embeddings of neurons can approximate the image embedding. Similarly, we assume that a neuron's embedding can be formed by collectively considering the embeddings of images to which the neuron strongly responds, In particular, we aim to encode a common concept (e.g., car wheel) across the stimuli (e.g, car images) into the neuron's embedding. To approximate a neuron's embedding, we consider linearly combining the embeddings of the stimuli of the neuron, reinforcing the common concepts (e.g., car wheel) by summing the shared features encoded in the image embeddings. Unrelated concepts (e.g., backgrounds or different colors of cars) which may occur randomly and vary in presence (or absence) across stim-

uli can be disregarded by summing and zeroing out such unrelated concepts' (positive and negative) contributions. To aggregate the embeddings, we adopt the standard practice of averaging across the important images as in previous seminar work [183, 62, 47]. Eq (7.4) presents the neuron embedding approximation, where $X^{t_b}_{n,M_b}$ is the set of stimuli of neuron n in the base model M_b at epoch t_b.

$$\mathbf{v}'^{t_b}_{n,M_b} = \frac{1}{|X^{t_b}_{n,M_b}|} \sum_{\mathbf{x} \in X^{t_b}_{n,M_b}} \mathbf{v}_{\mathbf{x}} \tag{7.4}$$

Eq (7.5) presents the objective function to minimize the difference between the original and the approximated embedding of neurons in the base model, where N_{M_b} is a set of all neurons in the base model. We randomly initialize the image embeddings and learn them by gradient descent. Eq (7.6) shows the derivative used to update an image's embedding, where $N_{M_b,\mathbf{x}}$ is the set of neurons in M_b whose stimuli includes an input $\mathbf{x}$.

$$J_2 = \frac{1}{2} \sum_{n \in N_{M_b}} \left|\left| \mathbf{v}'^{t_b}_{n,M_b} - \mathbf{v}^{t_b}_{n,M_b} \right|\right|^2_2 \tag{7.5}$$

$$\frac{\partial J_2}{\partial \mathbf{v}_{\mathbf{x}}} = \sum_{n \in N_{M_b,\mathbf{x}}} \frac{1}{|X^{t_b}_{n,M_b}|} \left(\mathbf{v}'^{t_b}_{n,M_b} - \mathbf{v}^{t_b}_{n,M_b} \right) \tag{7.6}$$

The image embedding approach may have a limitation as it can only represent images from the top-k stimuli of neurons in the base model. Consequently, if none of the images in a neuron's stimuli are not covered by the base model, the neuron itself remains unrepresented. With a large number of images, the top-k sets of stimuli for two models may have a low chance of overlapping. To address this issue, we use a randomly sampled images (10% sampled) instead of using all of them to increase the chance of overlapping. Additionally, we indirectly represent images that are not covered by the base model's stimuli by adopting a similar approach as in Step 1; instead of representing neurons based on their co-activation by common images, we represent images based on how they make common neurons co-activated. For each image $\mathbf{x}$, CONCEPTEVO identifies the k most activated neurons by $\mathbf{x}$, denoted as $N^{t_b}_{M_b,\mathbf{x}}$. Images $\mathbf{x}_1$ and $\mathbf{x}_2$ are paired if there are common neurons in $N^{t_b}_{M_b,\mathbf{x}_1}$ and $N^{t_b}_{M_b,\mathbf{x}_2}$. The paired images are added to the multiset of image pairs denoted as S. Image pairs in S may appear more than once (i.e., S is a multiset), indicating that those images can stimulate more common neurons, leading to a closer embedding. The image embeddings are learned in a similar manner to the neuron embedding approach, with the embeddings for images that are already represented by the base model being fixed.

Step 2.2: Approximating embedding of neurons in other models at different epochs. After embedding images in Step 2.1, CONCEPTEVO approximates neuron embeddings of other models at other epochs by averaging the embedding of images in each neuron's stim-

uli that are covered by the base model. If none of the images in a neuron's stimuli are covered by the base model, it averages the indirectly derived image embeddings. Step 2.2 is the only necessary (sub)step when projecting concepts in a new model onto the unified semantic space. There is no need to repeat Step 1 and Step 2.1.

To visualize the neuron embeddings, we use UMAP, a non-linear dimensionality reduction method that preserves both the global data structures and local neighbor relations [111]. To assist in understanding the concepts that neurons strongly respond to, we compute example patches which are cropped images that maximally activate the neuron (e.g., example patches of neurons for the *"dog face"* concept in Fig 7.1) [56].

7.2.3 Concept Evolutions Important for a Class

Our objective, as discussed in **D2**, is to uncover crucial concept evolutions that impact class predictions. For example, how important is the evolution of a neuron's concept (e.g., from *"furry animals' eyes"* to *"human neck"*) to the prediction for a class (e.g., "bow tie")? Inspired by [47], we quantify the significance of a concept evolution by evaluating how sensitive a class prediction is to the evolutionary state of the concepts.

Eq (7.8) defines such sensitivity of the class c prediction with respect to the concept evolution of neuron n in layer l in model M, from epoch t to t', given an input $\mathbf{x}$. $Z^t_{l,M}(\mathbf{x})$ is the activation map of all neurons in l at t for $\mathbf{x}$. The function $h^t_{l,M,c}(\cdot) : \mathbb{R}^{h_l \times w_l \times s_l} \rightarrow \mathbb{R}$ takes $Z^t_{l,M}(\mathbf{x})$ as input and provides the logit value for class c, where h_l, w_l, and s_l are height, width, and the number of neurons in l, respectively. $\Delta Z^{t,t'}_{n,l,M}(\mathbf{x})$ is the activation change of n from t to t', as defined in Eq (7.7), where $\mathbf{0}_{a,b}$ is a zero matrix of a rows and b columns. The directional derivative in Eq (7.8) indicates how sensitively a prediction for class c would change if the activation in layer l changes towards the direction of neuron n's evolution. A positive value indicates that the concept evolution of neuron n positively contributes to the prediction for class c.

$$\Delta Z^{t,t'}_{n,l,M}(\mathbf{x}) = [\mathbf{0}_{h_l,w_l}, \cdots, \underbrace{Z^{t'}_{n,l,M}(\mathbf{x}) - Z^{t}_{n,l,M}(\mathbf{x})}_{n\text{-th matrix}}, \cdots, \mathbf{0}_{h_l,w_l}] \tag{7.7}$$

$$\begin{aligned} S^{t,t'}_{n,l,M,c}(\mathbf{x}) &= \lim_{\epsilon \rightarrow 0} \frac{h^t_{l,M,c}\big(Z^t_{l,M}(\mathbf{x}) + \epsilon \Delta Z^{t,t'}_{n,l,M}(\mathbf{x})\big) - h^t_{l,M,c}\big(Z^t_{l,M}(\mathbf{x})\big)}{\epsilon} \\ &= \nabla h^t_{l,M,c}(Z^t_{l,M}(\mathbf{x})) \cdot \Delta Z^{t,t'}_{n,l,M}(\mathbf{x}) \end{aligned} \tag{7.8}$$

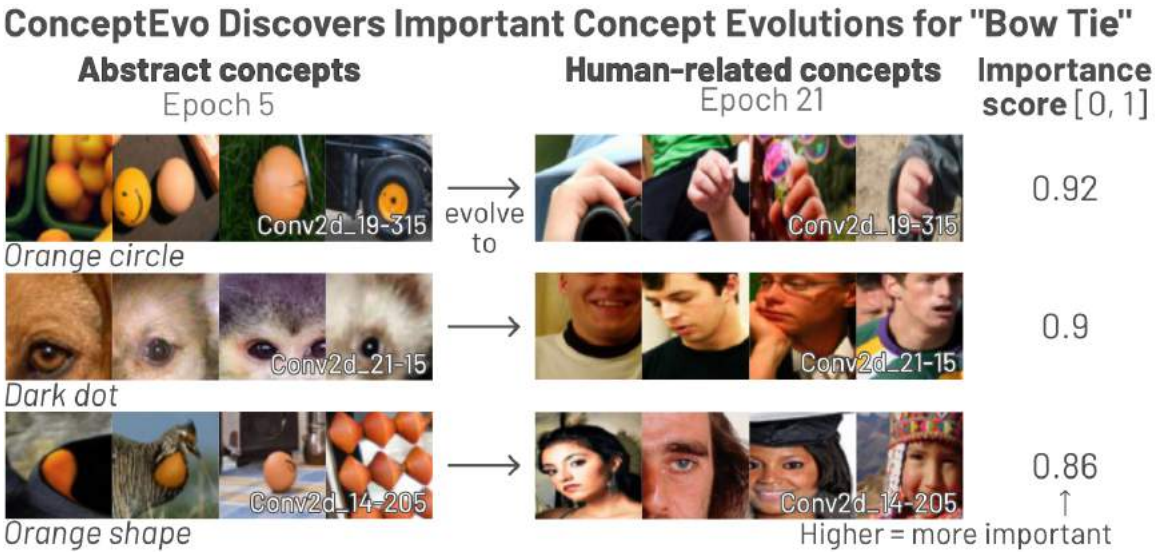


Figure 7.3: CONCEPTEVO identifies and quantifies important concept evolutions for class prediction. For example, in a VGG16, it discovers that concepts evolving towards human-related attributes, such as *"orange circles"* $\rightarrow$ *"hand"* in the top row, are important for the "bow tie" class. The importance score for this evolution is 0.92, meaning that such a concept evolution enhances predictions for 92% of bow tie images.

We finally measure the importance of concept evolution of a neuron n in layer l in model M from epoch t to t' for class c, by aggregating the importance across class c images, as in Eq (7.9), where X_c is the set of images labeled as c.

$$I^{t,t'}_{n,l,M,c} = \frac{|\{\mathbf{x} \in X_c \ : \ S^{t,t'}_{n,l,M,c}(\mathbf{x}) > 0\}|}{|X_c|} \tag{7.9}$$

Fig 7.3 illustrates important concept evolutions for the "bow tie" class discovered by CONCEPTEVO, such as evolutions from abstract concepts to *"hand," "neck,"* and *"face"* concepts. Surprised by the many evolutions towards human-related concepts, we inspected the raw images for the bow tie class and found that the majority of the images (over 70%) depict a person wearing a bow tie.

7.2.4 Runtime and Time Complexity

We designed CONCEPTEVO with a focus on practicality, considering the need for real-time interpretation during model training. To ensure this, we aimed to keep the runtime of our approach shorter than a single training epoch, allowing simultaneous training and interpretation. Our approach meets this requirement. Below, we report the runtime of CONCEPTEVO when using an NVIDIA A6000 GPU with 40GB RAM and the 10% randomly sampled ImageNet dataset [51] with 120 K images.

In the two-step concept evolution interpretation method of CONCEPTEVO (described in Sec. 7.2.2), Step 1, which creates the base semantic space, completes in less than 30 minutes. Step 2, which unifies the semantic space of models across epochs, takes less than 3 hours for Step 2.1 (image embedding) and less than 1 hour for Step 2.2 (identifying stimuli of a non-base model and approximating the embedding of its neurons).

Step 2.2 ($\sim$1 hour) is the only procedure that needs to be performed when projecting concepts in a new model onto the unified semantic space, and its runtime is shorter than training a model for an epoch (e.g., ConvNeXt takes 1.56 hours). This means that CONCEPTEVO's interpretation can be performed concurrently with model training. Step 1 ($\sim$30 minutes) and Step 2.1 ($\sim$3 hours) are one-time computations that can be reused, making CONCEPTEVO a practical and efficient choice.

General Interpretation of Concept Evolution (Sec 7.2.2). Now, we provide a detailed analysis of the time complexity of CONCEPTEVO's two-step concept evolution interpretation method described in Section 7.2.2. The "steps" mentioned here correspond to the steps outlined in Section 7.2.2.

Step 1: Creating the base semantic space. Step 1 has an overall time complexity of $O(|N_{M_b}| \cdot |I|)$, where N_{M_b} is the set of neurons in M_b, and I is the set of images.

In Step 1.1, the time complexity is $O(|N_{M_b}| \cdot |I|)$. For each neuron, collecting the top k images from $|I|$ images takes $O(|I| \cdot k)$. This process involves maintaining a sorted list of length k, which stores the top-k images observed so far. At each iteration for an image $\mathbf{x}$, we compare $\mathbf{x}$ to the smallest top-k item in the list. If $\mathbf{x}$ results in a higher activation for the neuron, we insert $\mathbf{x}$ into the list and remove the previous smallest top-k item. Identifying the proper spot to insert $\mathbf{x}$ and inserting it (if necessary) takes $O(k)$, and k is small (e.g., 10). Thus, the total time for collecting the top k images from $|I|$ images is $O(|I| \cdot k) = O(|I|)$. Therefore, for all neurons, Step 1.1 has a time complexity of $O(|N_{M_b}| \cdot |I|)$.

In Step 1.2, the time complexity is $O(|N_{M_b}|)$. Step 1.2 consists of two sub steps. First, for each image $\mathbf{x}$, collecting neurons with $\mathbf{x}$ in their stimuli takes $O(|N_{M_b}|)$, as it requires iterating through all stimuli of all neurons, which is a total of $O(k \cdot |N_{M_b}|)$. Second, for each image $\mathbf{x}$ and its corresponding co-activated neurons, sampling neuron pairs from the list of co-activated neurons with the sliding window takes $O(k \cdot |N_{M_b}|) = O(|N_{M_b}|)$. This results in $O(|N_{M_b}|)$ pairs of neurons. The sampling process is repeated E times, thus the total time for Step 1.2 is $O(E \cdot (|N_{M_b}| + |N_{M_b}|)) = O(|N_{M_b}|)$.

Step 1.3 takes $O(|N_{M_b}|)$, as the number of generated neuron pairs in Step 1.2 is $O(|N_{M_b}|)$. One epoch of gradient descent in Step 1.3 takes $O(|N_{M_b}| \cdot R) = O(|N_{M_b}|)$, resulting in a final time complexity of $O(|N_{M_b}|)$.

Overall, the time complexity of Step 1 is $O(|N_{M_b}| \cdot |I|)$ + $O(|N_{M_b}|) + O(|N_{M_b}|)$ = $O(|N_{M_b}| \cdot |I|)$. One advantage of this approach is its linear time complexity with respect to the number of neurons, instead of quadratic time. This is because it avoids the need to compare and represent concepts for all pairs of neurons, and instead focuses on sampled

pairs of neurons.

Step 2: Unifying the semantic space. Overall, Step 2 has a time complexity of $O(|N_{M_b}| \cdot |I|)$. In Step 2.1, the time complexity is $O(|N_{M_b}| \cdot |I|)$. This is because optimizing J_2 takes $O(|I|)$ time to learn $O(|I|)$ vectors, and approximately representing images not covered by the base model's stimuli also takes $O(|N_{M_b}| \cdot |I|)$, similar to Step 1 (since it adapts Step 1). To represent the concepts of neurons in a non-base model M within the unified semantic space, Step 2.2 takes $O(|N_M| \cdot |I|)$. This step involves computing stimuli for each neuron in M, where N_M is the neurons in M, using a similar approach as in Step 1.1.

Concept Evolution Important for a Class (Sec. 7.2.3). Identifying important concept evolutions for each class c entails a computational time complexity of $O(|I| \cdot |N_{M_b}|)$, since the computation of neuron sensitivity (Eq (7.8)) relies on the number of images labeled as c (which is $O(|I|)$). In terms of runtime, on average, this process took 37 minutes for VGG16, InceptionV3, and ConvNeXt models.

7.3 Experiment

We evaluate how well CONCEPTEVO satisfies the desired properties for interpreting concept evolution (Sec 7.2.1, D1-3) by addressing the following research questions:

- **Q1 Alignment.** How effectively does CONCEPTEVO align concepts of different models at different training stages in the unified semantic space? (Sec 7.3.2, for **D1**)
- **Q2 Meaningfulness.** To what extent are the discovered concept evolutions semantically meaningful? (Sec 7.3.3, for **D1**)
- **Q3 Importance.** How important are the discovered concept evolutions in terms of their impact on class prediction? (Sec 7.3.4, for **D2**)
- **Q4 Discoveries.** How does CONCEPTEVO contribute to the discovery of insightful findings? (Sec 7.3.5, for **D3**)

7.3.1 Experiment Settings

Datasets and models. We examine concept evolutions in representative image classifiers trained on ILSVRC2012 (ImageNet) [51]. The models we investigate include a modern model, such as ConvNeXt [187] which draws inspiration from recent architectures such as ResNet [188], ResNeXt [189], and vision transformers [190, 191]. Additionally, we investigate classic models such as VGG16 [184], VGG19 [184], VGG16 without dropout layers [192], and InceptionV3 [87]. To ensure comparable accuracies, we trained these models using the hyperparameters reported in prior work [184, 87, 187].

Hyperparameter settings. To create a unified semantic space that optimally balances coherence among neighboring neurons with computational efficiency, we carefully selected the following hyperparameters within their respective test ranges: the number of stimuli per

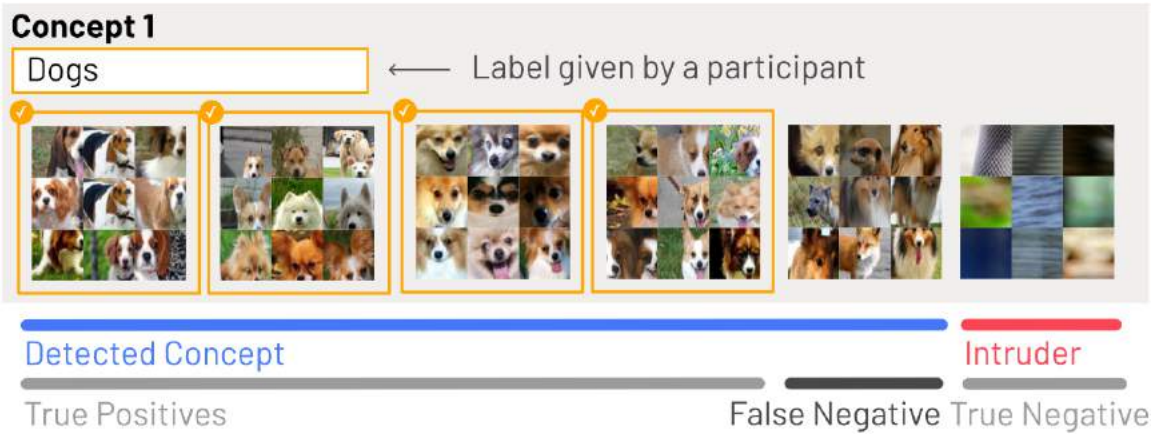


Figure 7.4: MTurk questionnaire example. Participants are presented with six neurons' example patches and asked to determine if they are a semantically coherent group. If they identify a coherent group, they provide a short label for that group. In the provided example, the first five neurons are semantically similar, detected and grouped by CONCEPTEVO. The rightmost is randomly sampled and unrelated to others. Here, a participant correctly identifies the first four neurons as a coherent "dogs" concept (four *true positives*), misses the fifth neuron (one *false negative*), and correctly identifies the intruder as unrelated (one *true negative*).

neuron (k) was tested from 5 to 30, with a chosen value of 10 for balancing the coherence and efficiency; the dimension of neuron and image embeddings was set to 30 (tested from 5 to 100); the learning rate for neuron embedding was set to 0.05 and for image embedding, it was set to 0.1 (tested from 0.001 to 0.5); and the number of randomly sampled neurons per neuron pair (R) was set to 3 (tested from 0 to 5).

7.3.2 Alignment of Neuron Embeddings

To ensure the effectiveness of CONCEPTEVO in aligning concepts across models and epochs, we conducted a large-scale human evaluation using Amazon Mechanical Turk (MTurk), following the methodology of prior work [80, 62]. The evaluation focused on four categories: (1) hand-picked sets of neurons representing similar concepts, which served as a baseline; (2) neuron groups detected by CONCEPTEVO from the base model (a well-trained VGG16); (3) neuron groups in the same model at different training epochs, detected by CONCEPTEVO; (4) neuron groups from different models at different epochs, detected by CONCEPTEVO. To collect the neuron groups, we applied K-means clustering on the neuron embeddings within the unified semantic space.

We conducted concept classification tasks with 260 MTurk participants, where each participant completed nine unique tasks. In each task, participants were presented with six neurons presented in a random sequence. Five of these neurons had similar concepts that are either identified by CONCEPTEVO or were hand-picked, while one neuron served as a

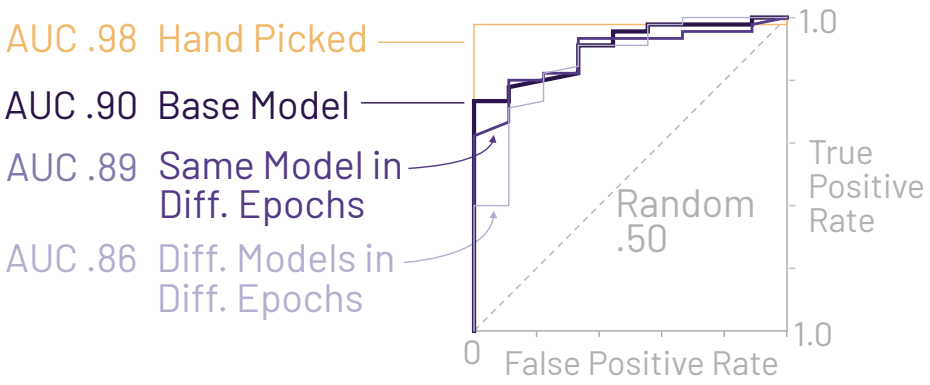


Figure 7.5: The ROC Curve, illustrating human estimations, demonstrates the notable alignability of concepts identified by CONCEPTEVO, consistently observed across various models and epochs.

randomly chosen "intruder" neuron. To help participants understand the concept of each neuron, we provided nine example image patches. Participants were not informed about the potential presence of intruders and were asked to select as many neurons as they believed to be semantically similar. They were also asked to provide a brief description of the concept they perceived. This process, as illustrated in Fig 7.4, essentially forms a classification task, treating the participants as classifiers and the grouped neurons as true labels. A total of 10,950 individual classification tasks were generated for the test set. From this framing, we consider success based on the level of agreement of participants with the model's determination. Fig 7.5 shows an ROC curve with the participants' determinations, demonstrating the high discernibility and alignment of CONCEPTEVO-detected concepts. Even when sampling concepts across different epochs and models, the AUC scores remain consistently high, ranging from 0.90 for sampling within the base model to 0.86 for sampling across different models and training epochs.

7.3.3 Meaningfulness of Concept Evolution

Concepts discovered by CONCEPTEVO should be meaningful and informative to humans. We evaluate the *interpretive consistency* of the concepts labeled and described by the participants, as shown in Fig 7.4. To handle variations in phrasing for the labels, we use sentence-level embeddings from the Universal Sentence Encoder (USE) [115]. USE captures the semantic similarity between phrases, such as "vehicle wheels," "cars," and "trucks", which should have high USE similarity. To establish a baseline for similarity, we calculate the average pairwise similarity between all labels, resulting in a value of 0.28. Subsequently, we measure the average pairwise similarity between the labels provided by participants for individual concepts within each category from 7.3.2. The results are as follows: (1) the average concept similarity for hand-picked concepts is 0.455, (2) the average concept similarity for concepts from the base model is 0.40, (3) the average concept similarity for

concepts within the same model but different epoch is 0.40, and (4) the average concept similarity for concepts from different models and different epochs is 0.38. All of these values significantly exceed the baseline similarity value of 0.28. This indicates that the concepts discovered through CONCEPTEVO are reliable and meaningful, even when assessed by different people.

7.3.4 Concept Evolutions Important to a Class

CONCEPTEVO quantifies and identifies important concept evolutions, as seeb in Fig 7.6. In InceptionV3, it reveals concept evolutions from abstract to bird-specific concepts, crucial for classifying "Goldfinch." Similarly, in ConvNeXt, it discovers evolutions from abstract concepts to dog-related concepts that are important for classifying the "Shetland sheepdog" class. As training progresses, some neurons become more specialized. For example, in the first row of Fig 7.6, a neuron initially detecting abstract concepts of a *dark background* evolves to detect a *dark-eyed circle* and later to detect a *head with a dark eye*. Appendix B provides additional examples of concept evolutions that are important to class predictions.

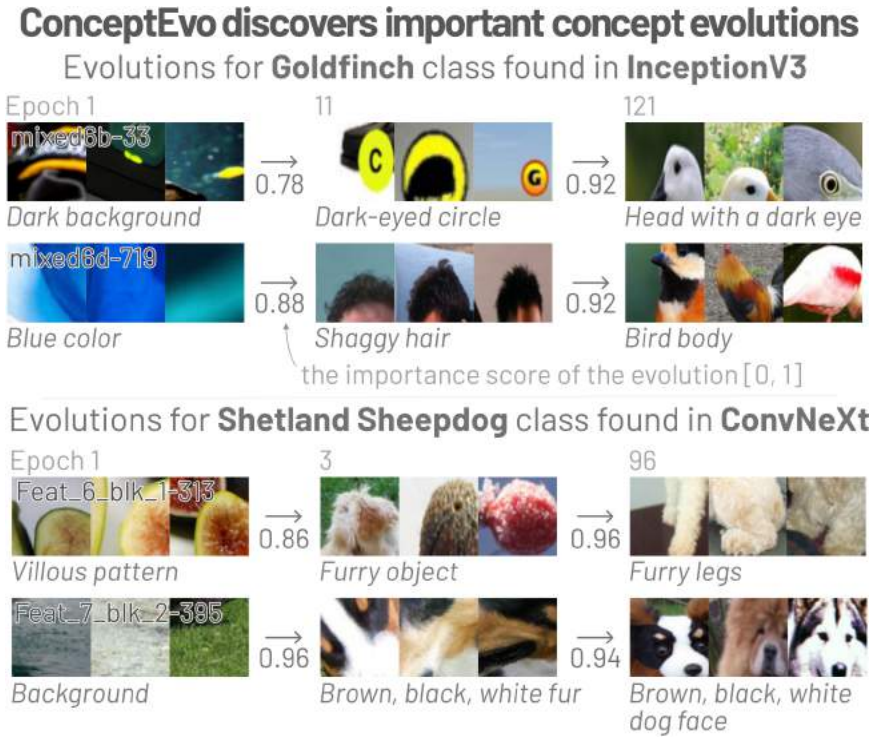


Figure 7.6: CONCEPTEVO discovers concept evolutions important for class predictions. For example, it discovers bird-related evolutions important for the "Goldfinch" class in InceptionV3, and dog-related evolutions important for the "Shetland sheepdog" class in ConvNeXt. Some neurons become increasingly specialized as training progresses. For example, in the first row, a neuron that initially detects abstract concept of *dark background* evolves to detect *dark-eyed circle*, and then further evolves to detect *head with a dark eye*.

To evaluate the effectiveness of CONCEPTEVO in discovering important concept evolutions, we measure the changes in accuracy when evolutions are reverted, similar to how prior work evaluated concept importance in fully-trained models [183, 62]. By reverting a neuron's activation map from t' to t, we evaluate the prediction accuracy at t'. A larger drop in accuracy indicates a higher importance for the concept evolution of that neuron. To determine the stages of evolution to evaluate, we identify the epochs with the closest top-1 training accuracies to the milestones of 25%, 50%, and 75%. Specifically, for VGG16, the evolution stages are 5→21 and 21→207; for InceptionV3, 1→11 and 11→121; and for ConvNeXt, 1→3 and 3→96.

As CONCEPTEVO measures concept evolution's importance for a single neuron (Eq 7.9), it is natural to evaluate accuracy changes by reverting each neuron's evolution individually and aggregating the changes. However, due to the large number of neurons, this approach is computationally prohibitive. Instead, we propose a more practical approach that reverts multiple evolutions in a layer at a time and aggregates the accuracy changes across layers. The evaluation process consists of five steps for each class c and evolution stage from epoch t to t'. **Step 1:** Sample 128 images for class c, which corresponds to approximately 10% of the total images for that class (∼1300 images). **Step 2:** Compute the importance of concept evolutions for all neurons, using Eq (7.9). **Step 3:** Rank the neurons in each layer based on their evolution importance and divide them into four importance bins: 0-25th percentile (most important), 25-50th percentile, 50-75th percentile, and 75-100th percentile. **Step 4:** Revert the evolutions of neurons in each bin, compute the accuracy at epoch t', and measure the accuracy changes compared to the non-reverted accuracy. **Step 5:** Average the accuracy changes across layers for the four bins. To mitigate sampling bias in Step 1, we repeat the procedure five times independently. We average the accuracy changes across 100 classes randomly chosen from ImageNet's 1,000 classes[2].

Fig 7.7 illustrates the impact of reverting evolutions in different importance bins on the top-1 training accuracy of VGG16, InceptionV3, and ConvNeXt. Notably, reverting higher-importance evolutions (lower percentiles) results in larger accuracy drops, confirming the effectiveness of CONCEPTEVO in quantifying and identifying important concept evolutions. Interestingly, reverting the least important evolutions (75-100th percentile) sometimes leads to increased accuracy. This suggests that the least important evolutions may interfere with the corresponding class predictions. As a baseline, we reverted 25% randomly selected evolutions, resulting in an accuracy drop between the 25-50th percentile and the 50-75th percentile. Furthermore, we evaluated the changes in the top-5 training, top-1 test, and top-5 test accuracies when reverting evolutions in the same four bins, reinforcing our key finding that reverting higher-importance evolutions results in a larger accuracy drop. Appendix C provides additional results for top-5 training, top-1 test, and top-5 test accuracy changes when some evolutions are reverted to a previous epoch.

[2]Standard deviations of the average accuracy changes across the classes between the five runs are very low (e.g., 9.2e-5 for top-1 training accuracy and 2.1e-4 for top-1 test accuracy, for the 21→207 evolution).

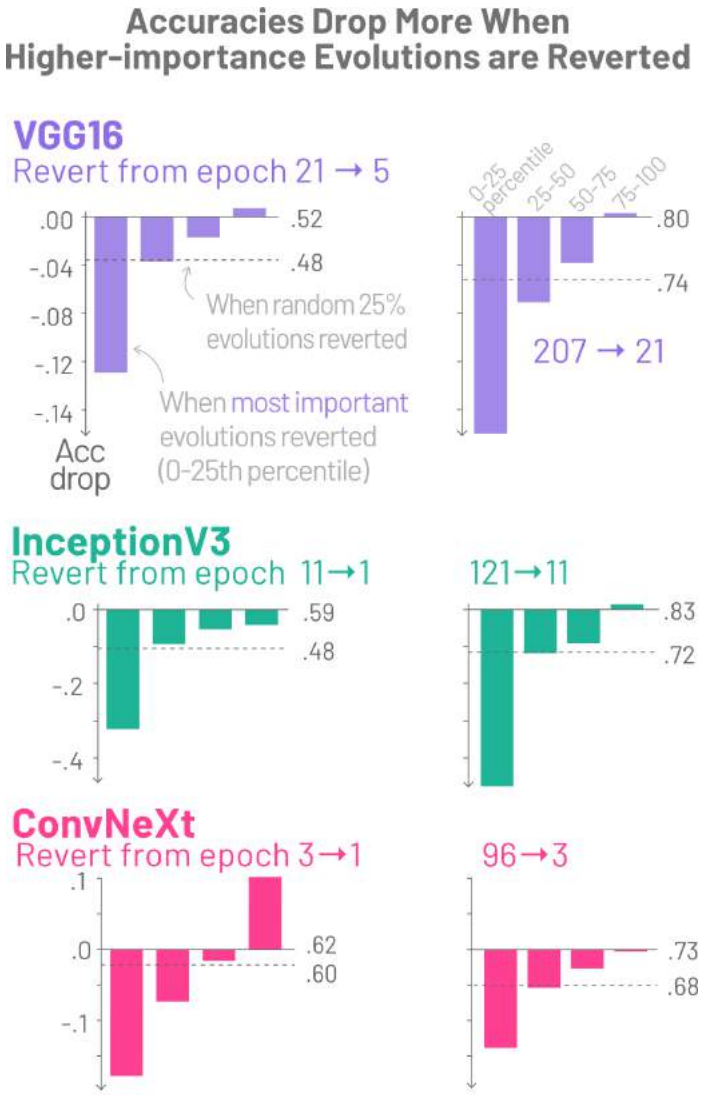


Figure 7.7: We evaluate the ability of CONCEPTEVO to quantify and identify important concept evolutions for 100 randomly selected classes. Neurons are ranked by their evolution importance score and then divided into four bins: 0-25th (top 25% most important), 25-50th, 50-75th, 75-100th percentiles. By reverting higher-importance evolutions, we observed a larger drop in top-1 training accuracy, demonstrating the effectiveness of CONCEPTEVO in quantifying and identifying important concept evolutions. As a baseline, for comparison, we also measured the accuracy drop when randomly reverting 25% (i.e., the same number of neurons in each bin) evolutions, which fell between the 25-50th and 50-75th percentile bins.

7.3.5 Discovery

Incompatible hyperparameters harm concept diversity. CONCEPTEVO's aligned neuron concept embedding helps identify problems caused by incompatible hyperparameters and offer insights into their impact on model performance. For example, in Fig 7.2b, CON-

Figure 7.8: An example of *"background"* concept detected by VGG16 and ConvNeXt that are trained with overly large learning rates, when the accuracy is very low. In the last convolutional layer in these models, a notable percentage (over 30%) of neurons show exclusive intense activation in response to backgrounds of images.

CEPTEVO reveals that a VGG16 suboptimally trained with an excessively high learning rate[3] exhibits a drastic accuracy drop over training epochs. Early signs of problems, such as the "atrophying" of neuron concepts that degrade concept diversity and only detect lower-level concepts, become apparent even before the accuracy reaches 0. The loss of diversity is so severe that it cannot be recovered even with 40 additional training epochs. A similar pattern is observed in a ConvNeXt model trained with a high learning rate[4], as shown in Fig 7.9a. In cases where the accuracy is low in VGG16 and ConvNeXt, we observe a significant reduction in concept diversity, especially in the last convolutional layers. For example, as seen in Fig 7.8, almost all neurons in VGG16 and over 30% of neurons in ConvNeXt predominantly detect *"background"* concepts.

In the case of an InceptionV3 unstably trained with a large learning rate[5], CONCEPTEVO reveals a similar yet slightly different scenario. As depicted in Fig 7.9b, the accuracy significantly drops at epoch 70, but interestingly, it recovers after a few more epochs. This recovery is likely due to the persistence of a large number of concepts at epoch 70 and the increasing diversity of concepts, despite the low accuracy.

These examples demonstrate that CONCEPTEVO can provide actionable insights to determine whether interventions, such as stopping the training, might be beneficial. Severe damage to concept diversity, as observed in Fig 7.2b and 7.9a, suggests that stopping the training might be more beneficial, as the model is unlikely to recover even with further epochs, compared to a better ability to recover the concept diversity as depicted in Fig 7.9b.

To quantitatively study concept diversity, we use differential entropy which measures the uncertainty in a continuous variable [193]. We compute the differential entropy for each dimension of neuron embeddings and average the values across the dimensions[6]. Higher values indicate more diverse concepts. In a VGG16 suboptimally trained with a large

[3] 0.05, larger than an optimal learning rate 0.01 presented in prior work

[4] 0.02, larger than an optimal learning rate 0.004 used in prior work

[5] 1.5, larger than an optimal rate of 0.045 used in prior work

[6] We average the differential entropy across reduced 2D embeddings, instead of the original dimension, since computing the differential entropy for some high dimensional vectors leads to infinity.

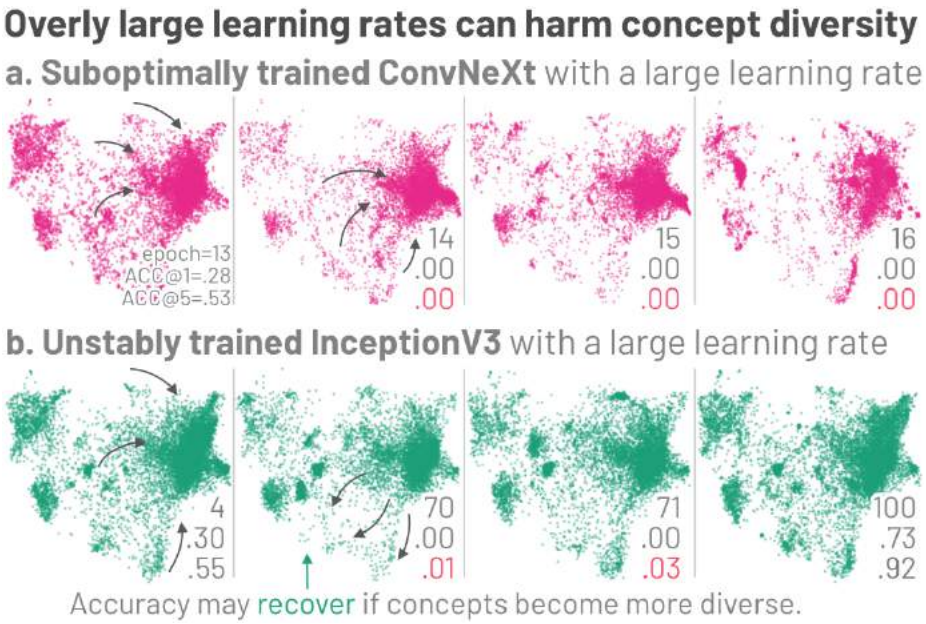


Figure 7.9: A suboptimally trained ConvNeXt and an unstably trained InceptionV3 with large learning rate experience decreased concept diversity and convergence in certain regions (e.g., right side to detect lower-level concepts), specifically when these models' training accuracies drop (as seen in the second column). Interestingly, the training accuracy of InceptionV3 recovers, because the concepts become more diverse starting from epoch 70, showing a better recovery resilience.

learning rate (Fig 7.2b), the differential entropy decreases: 1.89→1.48→-1.80→-1.80 for epochs 3, 12, 13, 14, indicating a loss of concept diversity. Similarly, in a suboptimally trained ConvNeXt (Fig 7.9a), the differential entropy decreases: 1.83→1.64→1.59→1.52 for epochs 13, 14, 15, 16. In contrast, optimally trained models show increasing differential entropy, indicating that concepts become more diverse over epochs. For example, in an optimally trained VGG16 (Fig 7.2a), the differential entropy increases: 1.10→1.90→2.06→2.09 for epochs 0, 5, 21, 207. In the case of an unstably trained InceptionV3 (Fig 7.9b), the differential entropy decreases until epoch 70 (lowest accuracy) and then rebounds: 1.82→1.32→1.54→1.80 for epochs 4, 70, 71, 100, indicating that its concept diversity was initially damaged but later restored.

Overfitting slows concept evolution. Overfitting is a common issue in DNN training [194, 195]. Using CONCEPTEVO, we have discovered that concepts in overfitted models evolve at a slower pace, despite experiencing rapid increases in training accuracy. To intentionally induce overfitting, we modified a VGG16 (Fig 7.2c) by removing its dropout layers which are known to help mitigate overfitting [192]. Additionally, we overfit a ConvNeXt model by setting the weight decay of the AdamW optimizer to 0, reducing its regularization effect [196]. These models are overfitted expectedly[7].

[7]In VGG16, at epoch 30, its top-1 train, top-5 train, top-1 test, top-5 test accuracies are 0.99, 1, 0.37, 0.61,

We observed that overfitted models show slower concept evolution compared to their corresponding well-trained models. To increase the top-1 training accuracy from approximately 0.25 to 0.5 and from approximately 0.5 to 0.75, the neuron embeddings in a well-trained VGG16 model (Fig 7.2a) move an average Euclidean distance of 2.08e-4 and 2.90e-4, respectively. In contrast, the overfitted VGG16 model (Fig 7.2b) exhibits much slower movement, with neuron embeddings only shifting by 1.94e-4 and 1.76e-4 for the same accuracy increments. Similarly, for the well-trained ConvNeXt model, raising the top-1 training accuracy from approximately 0.25 to 0.5 and from approximately 0.5 to 0.75 corresponds to neuron embeddings moving an average distance of 1.49e-4 and 1.33e-4, respectively. Conversely, the overfitted ConvNeXt model shows slower movement, with neuron embeddings shifting by only 1.48e-4 and 1.27e-4 for the same accuracy increments.

7.3.6 Comparison with Existing Approaches

We compare CONCEPTEVO with existing methods for representing evolving concepts. Existing methods are not optimized to capture changes across epochs; they can only be applied to one epoch at a time, independently of other epochs. In our comparison, we consider NeuroCartography [80] and ACE [62]. ACE represents concepts using image segments that activate a layer. We use the final layer to follow the approach described in the original work. For image segments, we use the Broden dataset [60]. For 2D visualization of concepts, we use UMAP [111]. To ensure alignment across epochs, we run UMAP for all epochs simultaneously, avoiding misalignment caused by independent epoch-based reduction.

The results show that CONCEPTEVO effectively aligns concepts across epochs, while existing methods exhibit misalignment. In Fig 7.10a, the "car-related" concept neurons consistently appear at the bottom in epochs 2, 5, and 207. In contrast, Fig 7.10b demonstrates that the "car-related" neurons represented by NeuroCartography exhibit flipping, rotation, and shifting across epochs. Similarly, Fig 7.10c shows that the "car-related" image segments represented by ACE exhibit significant shifting as the concept space changes during training.

7.4 Conclusion and Future Work

CONCEPTEVO is a unified interpretation framework for DNNs that reveals the inception and evolution of detected concepts throughout the training process. Through both large-scale human experiments and quantitative analyses, we have showcased the effectiveness of CONCEPTEVO in discovering concept evolutions that facilitate human interpretation of model training across different models. This framework not only aids in identifying

respectively. In ConvNeXt, at epoch 32, its top-1 train, top-5 train, top-1 test, top-5 test accuracies are 0.94, 0.99, 0.57, 0.80, respectively.

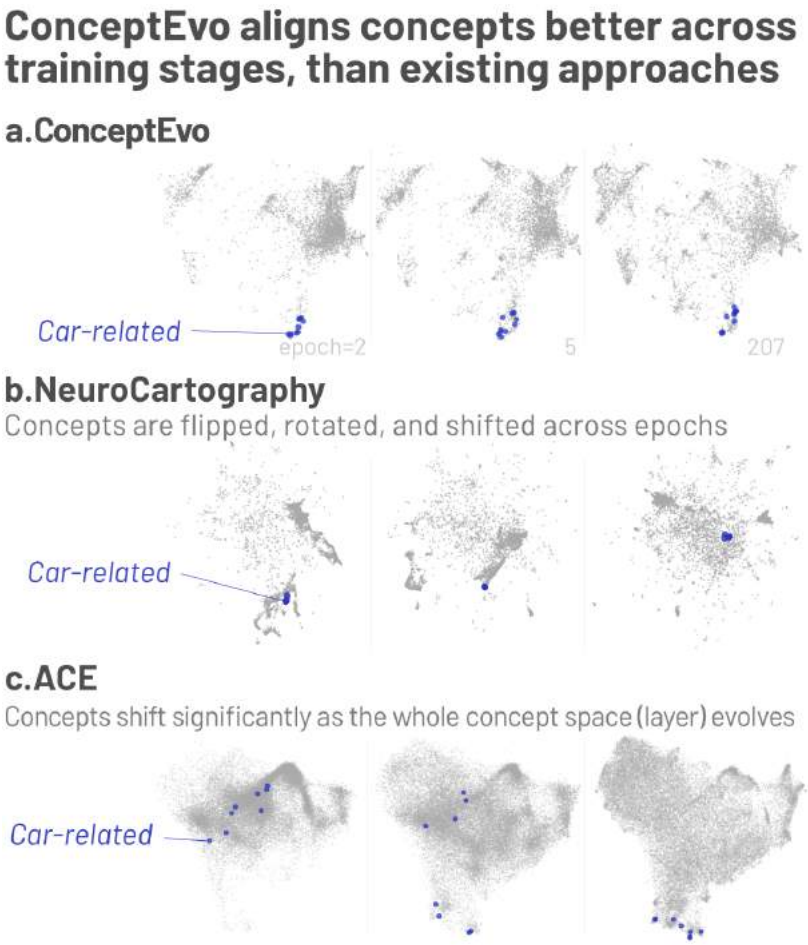


Figure 7.10: We compare the representation of concepts in VGG16 using ConceptEvo with existing methods. (a) The results show that CONCEPTEVO effectively aligns learned concepts across training epochs, by projecting similar concepts to similar embedding locations. (b) In contrast, concepts represented by NeuroCartography exhibit flipping, rotation, and shifting across epochs, indicating misalignment. (c) Similarly, concepts represented by ACE undergo significant shifting, as the entire concept space (layer activation space) changes during training, indicating misalignment as well.

potential training problems but also provides guidance for interventions to achieve more stable and effective training outcomes.

In our future work, we plan to expand the scope of our investigation to include other types of models, such as object detectors, reinforcement learning systems, and language models. Additionally, we aim to enhance the alignment of concepts across different models during training. Currently, our framework operates under the assumption that an image can be represented by linear combinations of various neurons. However, more complex relationships may exist beyond linear associations. Thus, we aspire to improve the concept alignment by considering these non-linear relationships, enabling a more comprehensive and accurate representation of concepts across different models.

Part IV

Conclusions and Future Directions

CHAPTER 8
CONCLUSIONS AND FUTURE DIRECTIONS

This thesis contributes scalable algorithms and novel interactive interfaces to interpret Deep Neural Networks (DNNs)—whether they are functioning optimally, sub-optimally, or evolving. Specifically, this thesis focuses on summarizing concepts learned within DNNs (Part I), revealing their vulnerabilities to adversarial attacks (Part II), and discovering the evolution of concepts during training process.

8.1 Research Contributions

This thesis makes research contributions to multiple fields, including interactive data visualization, machine learning, and more importantly their intersection.

New scalable systems for global, unified model interpretation.

- NEUROCARTOGRAPHY introduces two innovative scalable concept summarization techniques, *neuron clustering* and *neuron embedding*, to **automatically represent the vast conceptual space of all neurons in large DNNs and their relationships**, offering a holistic interpretation of large datasets like ImageNet with 1.2M images. By avoiding exhaustive comparisons between neuron pairs' concepts, our methods achieve linear time complexity with respect to the number of neurons, far surpassing the conventional quadratic time. A large-scale human evaluation with 244 participants shows that these concept summarization techniques discover coherent, human-meaningful concepts (Chapter 3).
- CONCEPTEVO is **the first unified framework for interpreting model training of multiple DNNs**, by refining our neuron embedding algorithm to synchronize the conceptual spaces from different models across training epochs into a unified semantic space. This alignment not only streamlines the comparative analysis of different model training processes, but also provides a powerful means to monitor model training and detects training anomalies. Our extensive human evaluation, involving 260 participants, demonstrates that our framework identifies concept evolutions that are both meaningful to humans and important for class predictions (Chapter 7).
- We develop BLUFF and SKELETONVIS, each a **first-of-its kind interactive system for deciphering attacks** to DNNs. BLUFF visualizes and compares the activation

pathways for benign and attacked images in vision-based neural networks (Chapter 5). SKELETONVIS visualizes and explains how attacks manipulate human joints in human action recognition models (Chapter 6).

Surprising discoveries and new insights.

- BLUFF sheds light on the previously unknown vulnerabilities of DNNs. Notably, we are the **first to visualize and characterize the variation in attack strategies based on attack intensity**. For instance, milder attacks may adopt a 'death by a thousand cuts' approach, while stronger attacks focus on exploiting a select few neurons. These discoveries can guide the creation of robust defensive measures, such as the elimination of susceptible neurons from the model. (Chapter 5)

- NEUROCARTOGRAPHY is among the **first system to map the comprehensive concept landscape of large DNNs, revealing unexpected phenomena**, such as the existence of isolated concept—a unique feature not tied to any particular image characteristics (e.g., "watermark" concept can appear on almost any kinds of images). Such surprising insights help guide model compression, suggesting the potential removal of such isolated neuron clusters. (Chapter 3)

- CONCEPTEVO **empowers users with new ways to identify potential model training issues** such as: (1) incompatible hyperparameters (e.g., overly high learning rate) severely harm concept diversity; and (2) concepts in overfitted models evolve slowly despite rapid training accuracy increases. Leveraging these insights allows for timely interventions, like halting training early when concept diversity is at risk. (Chapter 7)

Democratizing access to interpretability research through open-source systems.

- Prioritizing user convenience, our interactive visual systems, such as NEUROCARTOGRAPHY (Chapter 3), BLUFF (Chapter 5), and SKELETONVIS (Chapter 6), are presented as **interactive web applications**. These tools are universally accessible, compatible with any modern web browser, and free from the hassles of additional installations or the constraints of specific hardware.

- To foster a collaborative environment for DNN interpretability, we have made the **source codes of our algorithms available to the public**, including NEUROCARTOGRAPHY (Chapter 3), BLUFF (Chapter 5), and CONCEPTEVO (Chapter 7).

8.2 Impact

This thesis is making impact to academia, industry, and the government.

- NEUROCARTOGRAPHY (Chapter 3) has been highlighted as a top visualization publication (top 1% of 442 submissions) invited to present at SIGGRAPH.
- BLUFF (Chapter 5), SKELETONVIS, NEUROCARTOGRAPHY (Chapter 3), and CONCEPTEVO (Chapter 7) have been contributing to the multi-million dollar DARPA GARD (Guaranteeing AI Robustness against Deception) program in understanding model robustness and devising effective defenses.
- This dissertation has been recognized by a 2021 J.P. Morgan AI PhD Fellowship, among the only 15 awards in the world. Additionally, it was highlighted in 2022 Rising Stars in EECS, an international academic career workshop focusing on electrical engineering, computer science, and artificial intelligence.
- Research ideas developed in this dissertation contributed to multiple high-impact projects aimed at broadening the access to high-quality machine learning education. Among them are the NVIDIA Data Science Teaching Kit [24] now freely available to thousands of educators around the world, and CNN Explainer [25] that has been integrated into academic programs at institutions worldwide, such as Georgia Tech (Deep Learning), University of Wisconsin-Madison (Intro to AI), and University of Kyoto (Bioengineering).

8.3 Future Directions

While this dissertation has laid the groundwork for scalably and visually interpreting DNNs, it also unlocks new research directions along broadening the range of interpretable model types, protecting models from vulnerabilities, and optimizing the training process.

8.3.1 Expanding Interpretability to a Broader Spectrum of DNNs

This thesis focuses on interpreting the intricate workings of DNNs, that can serve as a foundational stepping stone towards broader horizons in other domains.

One immediate field for the application of our NEUROCARTOGRAPHY methodology is within specialized image domains, such as medical imaging. By adapting our techniques, we can summarize the concepts learned by DNNs, potentially revolutionizing fields such as radiology and providing more transparent DNN-powered diagnoses. This not only fortifies trust between medical professionals and machine learning models but also broadens the interpretability horizon.

Moreover, the insights gleaned from interpretation of image classification models through NEUROCARTOGRAPHY can be extended to cutting-edge image generative models, including the likes of Stable Diffusion [197] and DALL-E [198]. Given that NEUROCARTOG-

RAPHY offers visual representations of concept encoding by individual neurons, it is conceivable to adapt this for generative models. The focus would then shift to deciphering the encoding of text prompts within neurons and understanding the subsequent transformation of these encodings into image generation.

Looking beyond image models, our interpretation techniques hold promise for diverse data types, such as language models. As NEUROCARTOGRAPHY's concept summarization techniques are based on co-activation of neurons—rather than visual cues—we can decipher the intricate connections and activations in language models, leading to deeper insights into their linguistic processing dynamics.

8.3.2 Troubleshooting Models: Identifying and Shielding Vulnerable Neurons

DNNs, with their intricate architectures and expansive capabilities, present a compelling paradox within the machine learning community. While their advanced structure fuels unparalleled performance, it also shrouds them in the veil of interpretability challenges, leaving them vulnerable to both internal inconsistencies and external threats, such as adversarial attacks. Addressing these challenges calls for advanced tools capable of autonomously and interactively detecting, diagnosing, and resolving these issues. The depth of insights gleaned from this thesis can serve as the cornerstone for crafting such tools.

For example, when a DNN is subjected to adversarial attack, BLUFF can highlight the most vulnerable neurons and their connections — either being intensely excited or inhibited — thereby pinpointing the model's vulnerable pathways. Using the visual insights provided by BLUFF, we can envision a tool that empowers people to swiftly detect and eradicate these vulnerabilities. By selectively deactivating these susceptible neurons, users can potentially hinder the progression of threats deeper into the network, streamlining the troubleshooting process.

8.3.3 Real-time Monitoring and Steering of Network Training

In today's rapidly advancing world of deep learning, real-time monitoring and guidance during model training have become essential. Often, practitioners are left wondering if a model's training is progressing as expected or if there are issues that might impede its optimal learning.

Using CONCEPTEVO, we can establish clear benchmarks by studying common successful training patterns. This enables trainers to instantly determine if a model is on the right trajectory or if recalibrations are necessary. By doing so, we can conserve valuable time and resources, avoiding their use in unnecessary suboptimal training.

Furthermore, we can envision real-time training monitoring system that immediately alert trainers when models deviate from their expected training trajectory. These timely notifications prompt immediate actions such as adjusting training settings, redirecting train-

ing, or even pausing the training process. Such proactive, real-time monitoring ensures training remains efficient while maintaining the highest quality.

Additionally, we can consider a tool that allows DNN trainers to directly influence training in real-time. If the model is not picking up on certain important concepts, trainers can have the flexibility to modify the training data. This might include adding tougher challenges, such as adversarial examples, to address and reinforce areas where the model's understanding is lacking, ensuring a comprehensive understanding of those concepts.

8.4 Conclusion

My research exploration in interpreting DNNs' learned concepts, vulnerabilities, and evolutions has solidified a core belief: DNNs achieve their fullest potential not just through computational strength but when they align with human insights and trust. Thus, this thesis pushes for a human-focused perspective on DNN interpretability. I see a future where data-centric technologies enhance human intelligence, establishing a symbiotic relationship between humans and machines. This dissertation is an initial step towards this vision.

Appendices

APPENDIX A
NEURON EMBEDDING

We provide additional information on the neuron embedding algorithm employed in both NEUROCARTOGRAPHY (Chapter 3) and CONCEPTEVO (Chapter 7).

A.1 Formulating Neuron Embedding

We frame the neuron embedding algorithm as a maximum likelihood estimation problem: how can we learn neuron embeddings that optimally capture the conceptual similarity between neurons based on their co-activation? We formulate the likelihood of the similarity of concepts detected by two neurons n and m in a DNN M, as in Eq (A.1). In the equation, $\sigma(\cdot)$ is the sigmoid function (i.e., $\sigma(x) = 1/(1+e^{-x})$), and $\mathbf{v}_n$ is the embedding of n.

$$P(n, m) = \sigma(\mathbf{v}_n \cdot \mathbf{v}_m) \tag{A.1}$$

We define the likelihood objective function to maximize as in Eq (A.2). V is the set of neuron embeddings. D is the multiset of pairs of neurons that are strongly co-activated by many common images, as described in Chapter 7. r is a randomly-sampled neuron in M. R is the number of randomly-sampled neurons for each neuron pair (n, m). Intuitively, ① a pair of neurons with larger inner product of their embeddings has a higher likelihood of co-activation and higher similarity of their concepts, and ② randomly paired neurons are likely to have lower value of inner product and less likely to be conceptually similar.

$$\begin{aligned} P(D|V) &= \prod_{(n,m)\in D} \Bigg(\underbrace{P(n,m)}_{\substack{\text{① Co-activated} \\ \text{neuron pairs}}} \cdot \underbrace{\prod_{r=1}^{R} (1 - P(n,r))\,(1 - P(m,r))}_{\text{② Random neuron pairs}} \Bigg) \\ &= \prod_{(n,m)\in D} \Bigg(\underbrace{\sigma(\mathbf{v}_n \cdot \mathbf{v}_m)}_{\substack{\text{① Co-activated} \\ \text{neuron pairs}}} \cdot \underbrace{\prod_{r=1}^{R} (1 - \sigma(\mathbf{v}_n \cdot \mathbf{v}_r))\,(1 - \sigma(\mathbf{v}_m \cdot \mathbf{v}_r))}_{\text{② Random neuron pairs}} \Bigg) \end{aligned} \tag{A.2}$$

Based on Eq (A.2), we define the negative log-likelihood objective function J_1 to min-

imize, as in Eq (A.3). We randomly initialize the neuron embeddings and then learn the embeddings by using gradient descent that optimizes the objective function. For each pair of strongly co-activated neurons $(n, m) \in D$, we compute the derivative to update the neurons' embedding as in Eq (A.4) and (A.5).

$$J_1 = -\sum_{(n,m)\in D} \Bigg(\log\left(\sigma(\mathbf{v}_{n,M_b}^{t_b} \cdot \mathbf{v}_{m,M_b}^{t_b})\right) + \sum_{r=1}^{R} \log\left(1 - \sigma(\mathbf{v}_{n,M_b}^{t_b} \cdot \mathbf{v}_{r,M_b}^{t_b})\right) + \sum_{r=1}^{R} \log\left(1 - \sigma(\mathbf{v}_{m,M_b}^{t_b} \cdot \mathbf{v}_{r,M_b}^{t_b})\right) \Bigg) \tag{A.3}$$

$$\frac{\partial J_1}{\partial \mathbf{v}_n} = \left(1 - \sigma(\mathbf{v}_n \cdot \mathbf{v}_m)\right) \mathbf{v}_m - \sum_{r=1}^{R} \sigma(\mathbf{v}_n \cdot \mathbf{v}_r)\, \mathbf{v}_r \tag{A.4}$$

$$\frac{\partial J_1}{\partial \mathbf{v}_m} = \left(1 - \sigma(\mathbf{v}_n \cdot \mathbf{v}_m)\right) \mathbf{v}_n - \sum_{r=1}^{R} \sigma(\mathbf{v}_m \cdot \mathbf{v}_r)\, \mathbf{v}_r \tag{A.5}$$

Eq (A.6) and (A.7) present the derivative of the sigmoid and the log of sigmoid function used to calculate Eq (A.4) and (A.5).

$$\begin{aligned} \frac{d\sigma(x)}{dx} &= \frac{d\frac{1}{(1+e^{-x})}}{dx} \\ &= \frac{-(1+e^{-x})'}{(1+e^{-x})^2} \\ &= \frac{-(-e^{-x})}{(1+e^{-x})^2} \\ &= \frac{(e^{-x}+1-1)}{(1+e^{-x})^2} \\ &= \frac{1}{(1+e^{-x})} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\ &= \sigma(x) \cdot \big(1 - \sigma(x)\big) \end{aligned} \tag{A.6}$$

$$\begin{aligned}\frac{d\log(\sigma(x))}{dx} &= \frac{\sigma'(x)}{\sigma(x)} \\ &= \frac{\sigma(x)(1-\sigma(x))}{\sigma(x)} \qquad \cdots \text{by Eq (A.6)} \\ &= 1-\sigma(x)\end{aligned} \tag{A.7}$$

A.2 Pseudocode for Neuron Embedding

As outlined in the CONCEPTEVO (Chapter 7), the neuron embedding algorithm can be summarized in the following steps:

- Step A: Identifying stimuli
- Step B: Sampling frequently co-activated neuron pairs
- Step C: Learning the neuron embedding

We present a pseudocode of the neuron embedding algorihtm in Algorithm 1. For step A, it uses Algorithm 2 for computing stimuli for each neuron. For step B, it also uses Algorithm 3 for computing strongly co-activated neurons for each image and Algorithm 4 for sampling co-activated neuron pairs. Finally, for step C, it learns the neuron embedding, following Equation (A.4) and (A.5).

Algorithm 1 Neuron embedding

Input: $\mathbf{X}$: the set of all images, and hyperparameters: k, E, α, R
Output: V: the set of neuron embeddings

// Step A: Get stimuli of all neurons
N := the set of all neurons in M_b
X := a list of size $|N|$ to store stimuli for all neurons
for $n \in N$ **do**
  X_n = getStimuli(n, $\mathbf{X}$, k)
  $X[n] = X_n$
end for

// Step B: Sample neuron pairs
$D :=$ an empty multiset
for $\mathbf{x} \in \mathbf{X}$ **do**
  $N_{\mathbf{x}}$ = getStronglyCoActivatedNeurons($\mathbf{x}, X, N$)
  D = UpdateNeuronPairs(D, $N\mathbf{x}$)
end for

// Step C: Learn neuron embeddings
V = a list of randomly initialized neuron embeddings
for i in [1, ..., E] **do**
  for $(n, m) \in D$ **do**
    $\mathbf{v}_n = V[n]$
    $\mathbf{v}_m = V[m]$
    $\mathbf{v}_n = \mathbf{v}_n - \alpha \frac{\partial J_1}{\partial \mathbf{v}_n}$ (Eq (A.4))
    $\mathbf{v}_m = \mathbf{v}_m - \alpha \frac{\partial J_1}{\partial \mathbf{v}_m}$ (Eq (A.5))
    $V[n] = \mathbf{v}_n$
    $V[m] = \mathbf{v}_m$
  end for
end for

Return V

Algorithm 2 getStimuli

Input: n: a neuron, $\mathbf{X}$: the set of all images, and k: a hyperparameter for the length of stimuli

Output: X_n: the stimuli of n

X_n := an empty list of length k
A := a list of length k filled with -inf
l = the layer to which n belongs
for $\mathbf{x} \in \mathbf{X}$ **do**
  $Z_{n,l}(\mathbf{x})$ = activation map of n in l given $\mathbf{x}$
  $M = Max(Z_{n,l}(\mathbf{x})) \in \mathbb{R}$
  if $A[k-1] < M$ **then**
    i = the smallest index such that $M \geq A[i]$
    for $j \in [k-1, ..., i+1]$ **do**
      $A[j] = A[j-1]$
      $X_n[j] = X_n[j-1]$
    end for
    $A[i] = M$
    $X_n[i] = \mathbf{x}$
  end if
end for

Return X_n

Algorithm 3 getStronglyCoActivatedNeurons

Input: $\mathbf{x}$: an image, X: stimuli of all neurons, N: the set of all neurons

Output: $N_{\mathbf{x}}$: strongly co-activated neuron pairs for $\mathbf{x}$

$N_{\mathbf{x}}$:= an empty set
for neuron $n \in N$ **do**
  $X_n = X[n]$ (i.e., n's stimuli)
  if $\mathbf{x} \in X_n$ **then**
    $N_{\mathbf{x}} = N_{\mathbf{x}} \cup \{n\}$
  end if
end for

Return $N_{\mathbf{x}}$

Algorithm 4 UpdateNeuronPairs

Input: D: a previous set of neuron pairs, $N_{\mathbf{x}}$: strongly co-activated neuron pairs for $\mathbf{x}$

Output: D: the updated set of neuron pairs

RN = the randomly ordered $N_{\mathbf{x}}$
$s = |N_{\mathbf{x}}|$
for i in [0, ... $s - 1$] **do**
  $p = (RN[i], RN[i + 1])$
  $D = D \cup \{p\}$
end for

Return D

APPENDIX B

EXAMPLES OF CONCEPT EVOLUTION FOR CLASS PREDICTIONS

We provide additional examples for Section 7.2.3, presenting concept evolutions that are important to a class prediction. For example, Figure B.1, B.2, B.3, B.4, B.5, and B.6 show examples of concept evolutions in a VGG16 for classes "Shetland sheepdog," "Ladybug," "Payphone," "Oxcart," "Fire engine," and "Cassette." For each class, three important concept evolutions out of all evolutions that have the score larger than 0.8 are shown, which is in about top 0.5% according to the scores.

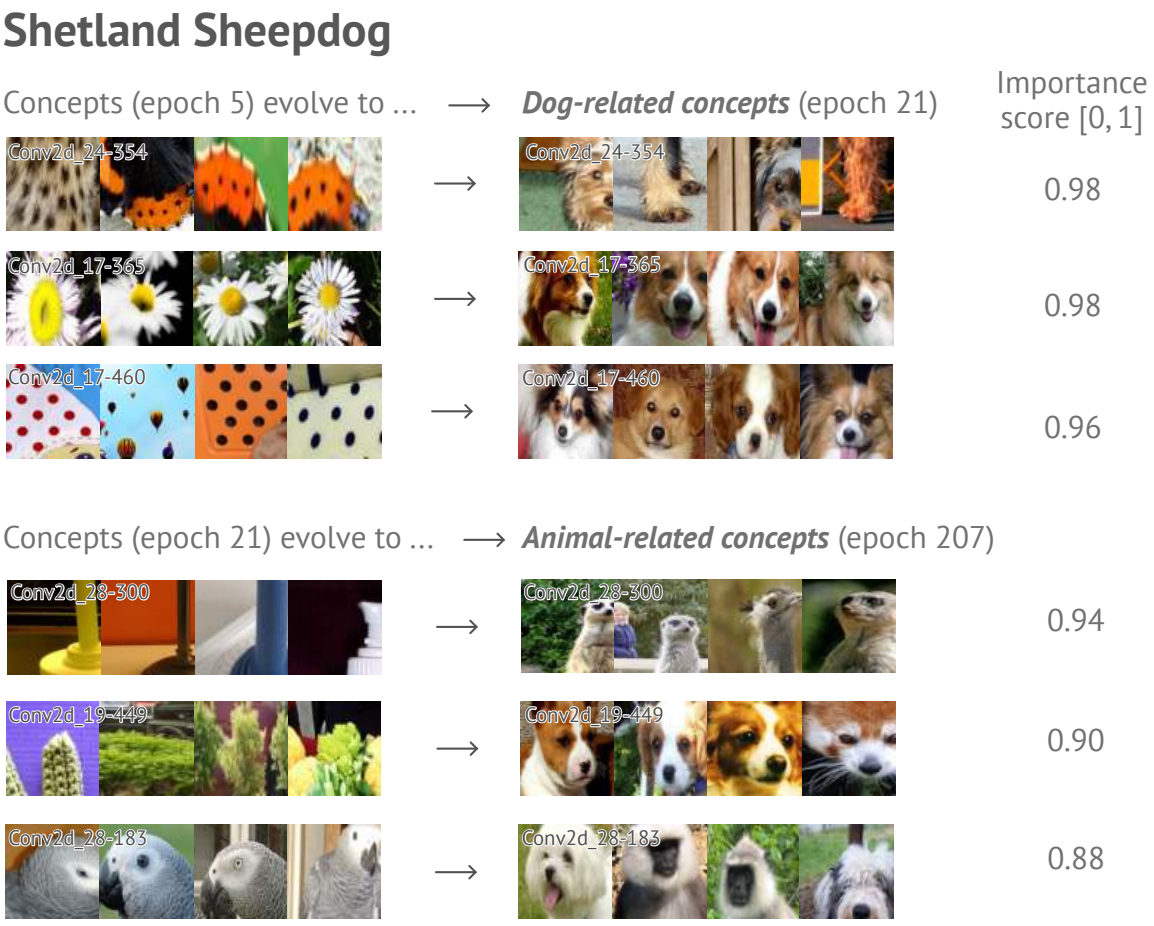


Figure B.1: Concept evolutions in a VGG16 important for “Shetland sheepdog” class.

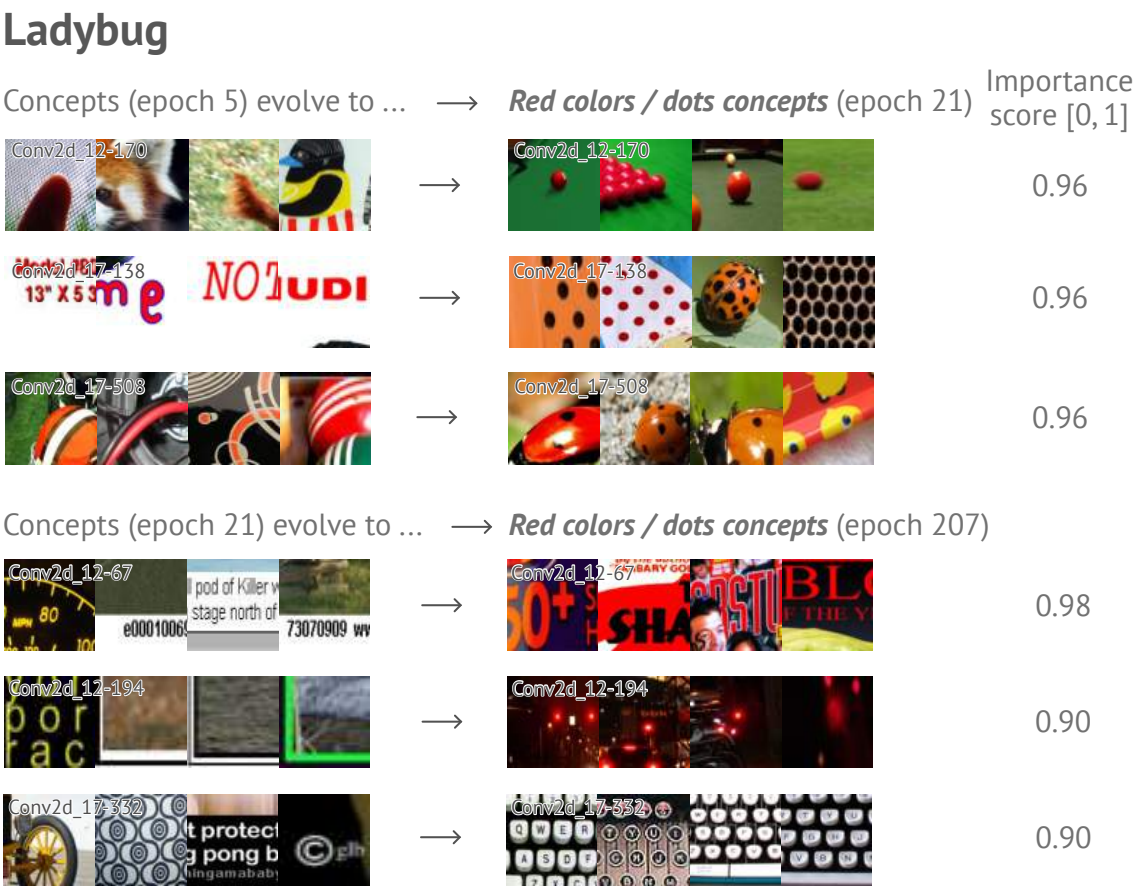


Figure B.2: Concept evolutions in a VGG16 important for “Ladybug” class.

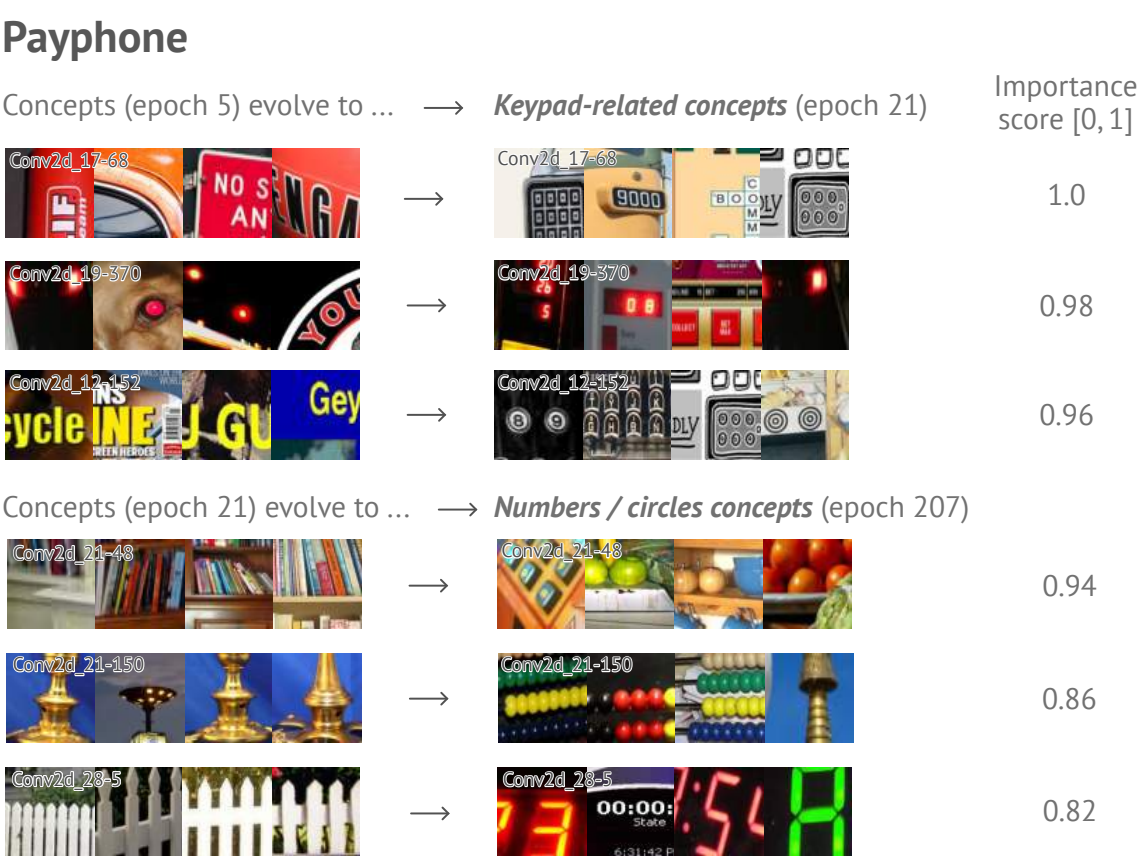


Figure B.3: Concept evolutions in a VGG16 important for “Payphone” class.

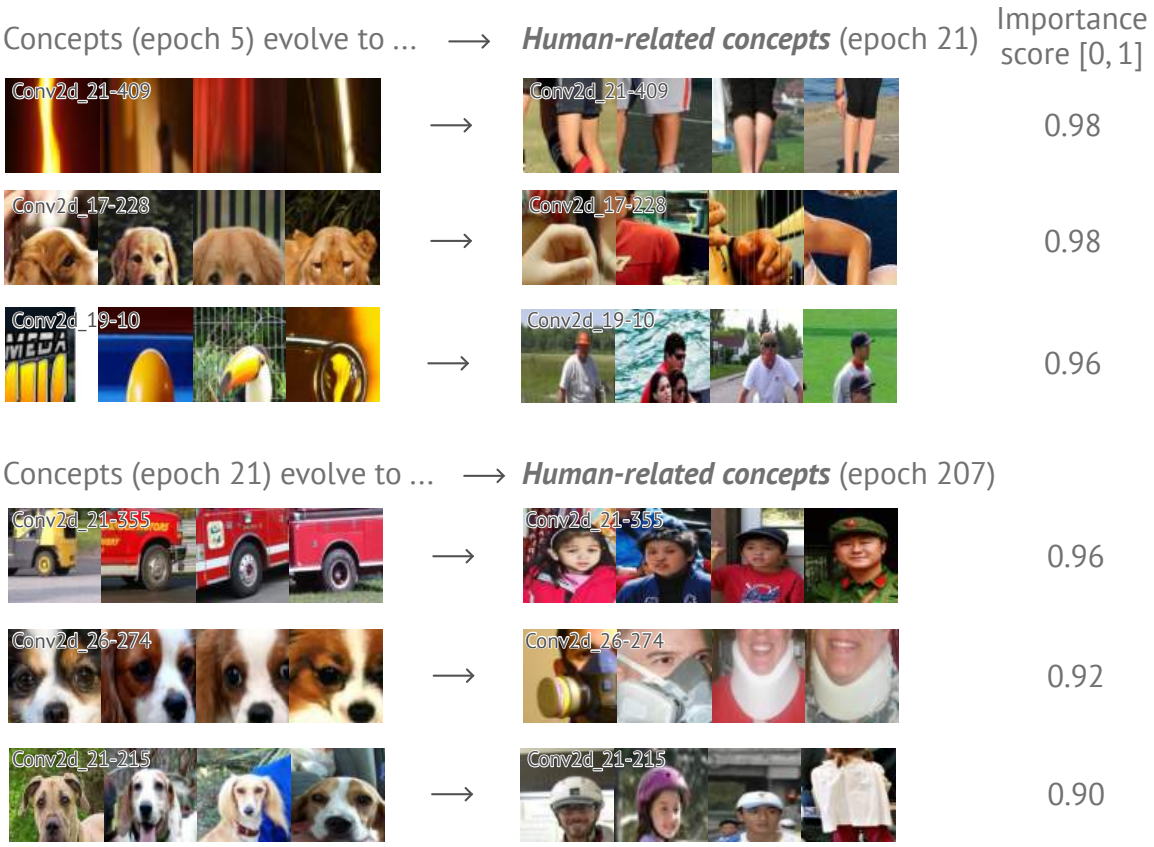


Figure B.4: Concept evolutions in a VGG16 important for “Oxcart” class.

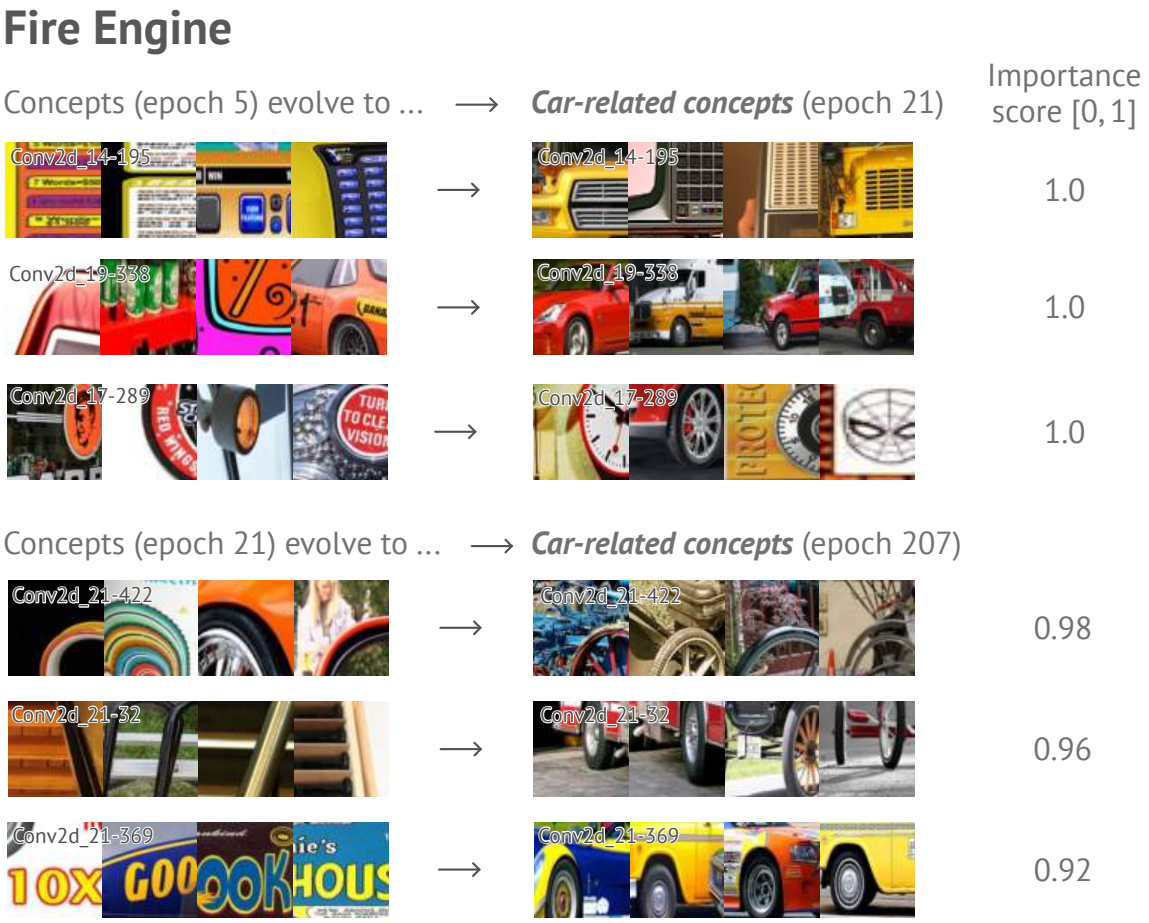


Figure B.5: Concept evolutions in a VGG16 important for “Fire engine” class.

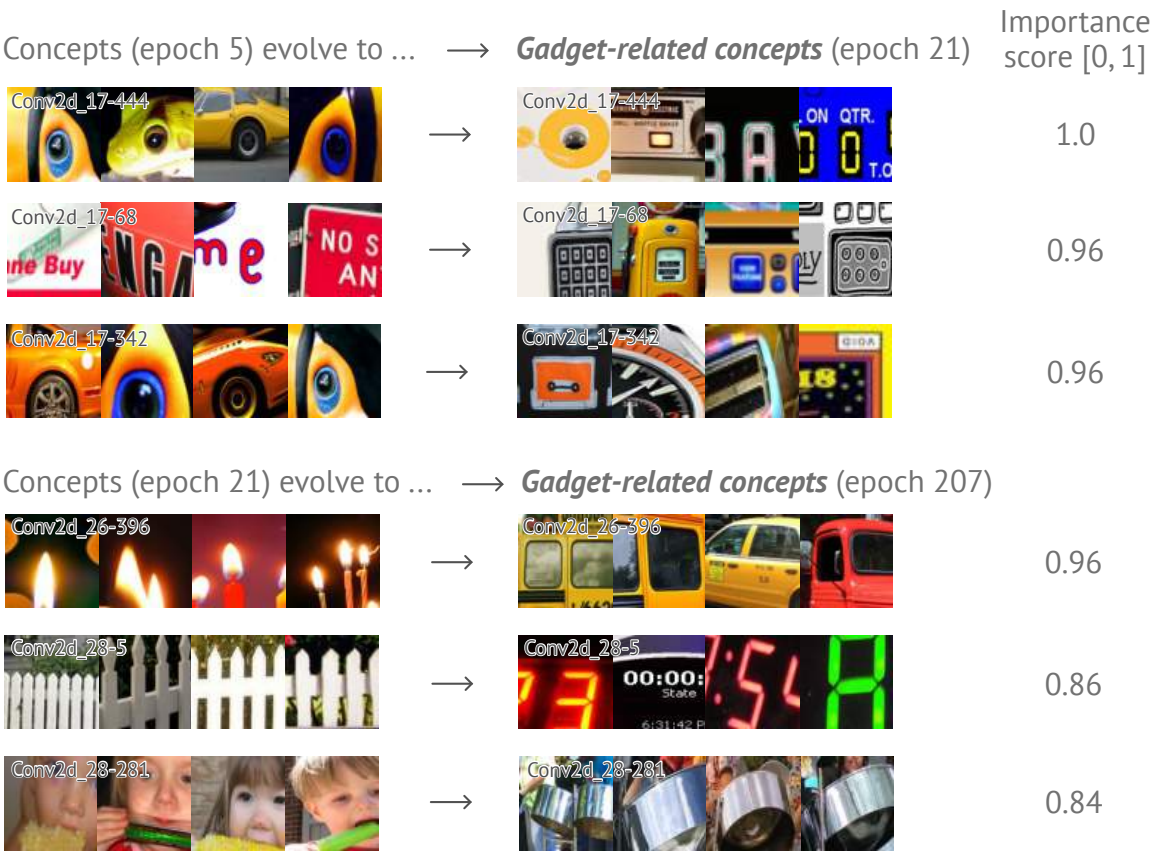


Figure B.6: Concept evolutions in a VGG16 important for “Cassette” class.

APPENDIX C

EVALUATION OF IMPORTANCE OF CONCEPT EVOLUTION

In Sec 7.3.4, we present the top-1 training accuracy changes when evolutions are reverted, which shows that reverting higher importance evolutions leads to greater accuracy drop. Figure C.1 provides additional results for top-5 training, top-1 test, and top-5 test accuracy changes when some evolutions are reverted to a previous epoch. We observe similar tendency as in the main paper: reverting higher importance evolutions leads to greater accuracy drop.

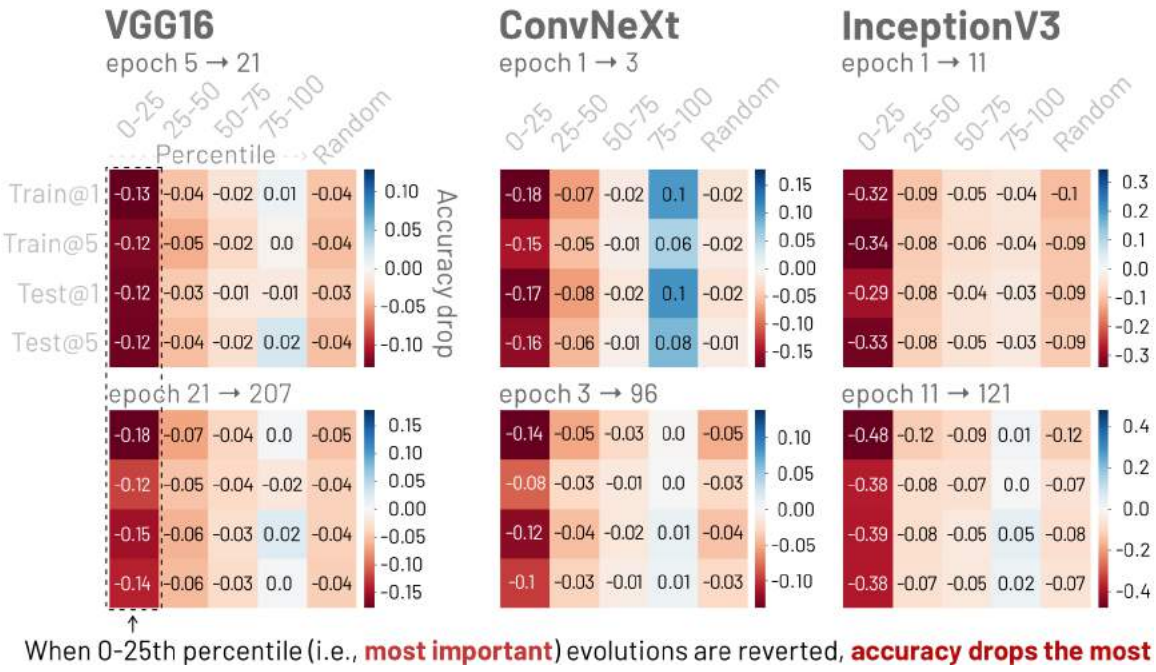


Figure C.1: Evaluation on how well CONCEPTEVO can find important concept evolutions for 100 random classes. We ranked neurons in the decreasing order of evolution importance computed by CONCEPTEVO, and placed them in 4 importance bins: 0-25th (most important), 25-50th, 50-75th, 75-100th percentiles. Reverting higher-importance evolutions leads to greater accuracy drop, confirming CONCEPTEVO's effectiveness in identifying important concept evolutions. For a baseline, we also computed the accuracy drop when 25% (i.e., the same number of neurons in each bin) of randomly selected evolutions were reverted, observing that the accuracy drop of random reversion is between that of 25-50th and 50-75th percentile bin.

REFERENCES

[1] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[2] V. Choutas, P. Weinzaepfel, J. Revaud, and C. Schmid, "Potion: Pose motion representation for action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7024–7033.

[3] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4651–4659.

[4] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6721–6729.

[5] J. Wan *et al.*, "Deep learning for content-based image retrieval: A comprehensive study," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 157–166.

[6] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th international conference on control automation robotics & vision (ICARCV)*, IEEE, 2014, pp. 844–848.

[7] J. Latif, C. Xiao, A. Imran, and S. Tu, "Medical imaging using machine learning and deep learning algorithms: A review," in *2019 2nd International conference on computing, mathematics and engineering technologies (iCoMET)*, IEEE, 2019, pp. 1–5.

[8] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu, "Analyzing the training processes of deep generative models," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 77–87, 2017.

[9] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "The dangers of post-hoc interpretability: Unjustified counterfactual explanations," *IJCAI*, 2019.

[10] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[11] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Advances in neural information processing systems*, vol. 31, 2018.

[12] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[13] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *5th International Conference on Learning Representations, ICLR*, 2017.

[14] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *ICLR*, 2018.

[15] S. Arora, N. Cohen, N. Golowich, and W. Hu, "A convergence analysis of gradient descent for deep linear neural networks," *International Conference on Learning Representations (ICLR)*, 2019.

[16] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[17] J. Safarik, J. Jalowiczor, E. Gresak, and J. Rozhon, "Genetic algorithm for automatic tuning of neural network hyperparameters," in *Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything*, SPIE, vol. 10643, 2018, pp. 168–174.

[18] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2018, pp. 52–68.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, vol. abs/1412.6572, 2014.

[20] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*, Chapman and Hall/CRC, 2018, pp. 99–112.

[21] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *International conference on machine learning*, PMLR, 2019, pp. 5231–5240.

[22] A. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[23] G. Tao, S. Ma, Y. Liu, and X. Zhang, "Attacks meet interpretability: Attribute-steered detection of adversarial samples," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[24] NVIDIA, *Dli data science teaching kits*, https://blogs.nvidia.com/blog/2021/02/25/dli-data-science-teaching-kits/, 2021.

[25] Z. J. Wang *et al.*, "Cnn explainer: Learning convolutional neural networks with interactive visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1396–1406, 2020.

[26] 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.

[27] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[28] S. Lundberg and S.-I. Lee. "The github readme for SHAP: A unified approach to explain the output of any machine learning model." Accessed: September 1, 2023. ().

[29] S. Lundberg and S.-I. Lee. "Example notebook demonstrating how to use the model agnostic kernel shap algorithm to explain predictions from the vgg16 network in keras." Accessed: September 1, 2023. ().

[30] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[31] S. Lipovetsky and M. Conklin, "Analysis of regression in game theory approach," *Applied Stochastic Models in Business and Industry*, vol. 17, no. 4, pp. 319–330, 2001.

[32] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowledge and information systems*, vol. 41, pp. 647–665, 2014.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.

[34] M. Moradi and M. Samwald, "Post-hoc explanation of black-box classifiers using confident itemsets," *Expert Systems with Applications*, vol. 165, p. 113 941, 2021.

[35] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 180–186.

[36] Y. Zhang, K. Song, Y. Sun, S. Tan, and M. Udell, ""why should you trust my explanation?" understanding uncertainty in lime explanations," *ICML AI for Social Good Workshop*, 2019.

[37] D. Alvarez-Melis and T. S. Jaakkola, "On the robustness of interpretability methods," *ICML Workshop on Human Interpretability in Machine Learning*, 2018.

[38] E. Lee, D. Braines, M. Stiffler, A. Hudler, and D. Harborne, "Developing the sensitivity of lime for better machine learning explanation," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, SPIE, vol. 11006, 2019, pp. 349–356.

[39] S. Saito, E. Chua, N. Capel, and R. Hu, "Improving lime robustness with smarter locality sampling," *Workshop on Adversarial Learning Methods for Machine Learning and Data Mining (AdvML)*, 2020.

[40] J. Petch, S. Di, and W. Nelson, "Opening the black box: The promise and limitations of explainable machine learning in cardiology," *Canadian Journal of Cardiology*, vol. 38, no. 2, pp. 204–213, 2022.

[41] D. Bau, "Dissection of deep neural networks," Ph.D. Thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2021.

[42] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *ICLR Workshop*, 2014.

[43] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: Removing noise by adding noise," *ICML Workshop on Visualization for Deep Learning*, 2017.

[44] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," *Advances in neural information processing systems*, vol. 31, 2018.

[45] 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.

[46] S. Gulshad and A. Smeulders, "Explaining with counter visual attributes and examples," in *Proceedings of the 2020 international conference on multimedia retrieval*, 2020, pp. 35–43.

[47] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, *et al.*, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," in *International conference on machine learning*, PMLR, 2018, pp. 2668–2677.

[48] A. M. Groen, R. Kraan, S. F. Amirkhan, J. G. Daams, and M. Maas, "A systematic review on the use of explainability in deep learning systems for computer aided diagnosis in radiology: Limited use of explainable ai?" *European Journal of Radiology*, p. 110 592, 2022.

[49] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[50] M. Ghassemi, L. Oakden-Rayner, and A. L. Beam, "The false hope of current approaches to explainable artificial intelligence in health care," *The Lancet Digital Health*, vol. 3, no. 11, e745–e750, 2021.

[51] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[52] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, Springer, 2014, pp. 818–833.

[53] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[54] A. Nguyen, J. Yosinski, and J. Clune, "Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks," *ICML Visualization for Deep Learning workshop*, 2016.

[55] 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, 2016.

[56] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, vol. 2, no. 11, e7, 2017.

[57] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015.

[58] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, "Zoom in: An introduction to circuits," *Distill*, vol. 5, no. 3, e00024–001, 2020.

[59] C. Watanabe, "Interpreting layered neural networks via hierarchical modular representation," in *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part V 26*, Springer, 2019, pp. 376–388.

[60] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6541–6549.

[61] R. Fong and A. Vedaldi, "Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8730–8738.

[62] A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim, "Towards automatic concept-based explanations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[64] C. Szegedy *et al.*, "Intriguing properties of neural networks," *ICLR*, 2014.

[65] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*, Springer, 2019, pp. 52–68.

[66] C. Zhang *et al.*, "Interpreting and improving adversarial robustness of deep neural networks with neuron sensitivity," *IEEE Transactions on Image Processing*, vol. 30, pp. 1291–1304, 2020.

[67] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *Advances in neural information processing systems*, vol. 32, 2019.

[68] H. Zheng, Z. Zhang, H. Lee, and A. Prakash, "Understanding and diagnosing vulnerability under adversarial attacks," *arXiv preprint arXiv:2007.08716*, 2020.

[69] 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.

[70] 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.

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

[72] D. Smilkov, S. Carter, D. Sculley, F. B. Viégas, and M. Wattenberg, "Direct-manipulation visualization of deep networks," *ICML*, 2016.

[73] S. Chung, C. Park, S. Suh, K. Kang, J. Choo, and B. C. Kwon, "Revacnn: Steering convolutional neural network via real-time visual analytics," in *Future of interactive learning machines workshop at the 30th annual conference on neural information processing systems (NIPS)*, 2016.

[74] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "Deepeyes: Progressive visual analytics for designing deep neural networks," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 98–108, 2017.

[75] P. E. Rauber, S. G. Fadel, A. X. Falcao, 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, 2016.

[76] M. Li, Z. Zhao, and C. Scheidegger, "Visualizing neural networks with the grand tour," *Distill*, vol. 5, no. 3, e25, 2020.

[77] W. Zhong *et al.*, "Evolutionary visual analysis of deep neural networks," in *ICML Workshop on Visualization for Deep Learning*, 2017, p. 9.

[78] 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.

[79] A. P. Wright *et al.*, "A comparative analysis of industry human-ai interaction guidelines," *IEEE Visualization Conference, Workshop on Trust and Expertise in Visual Analytics (TREX)*, 2020.

[80] H. Park *et al.*, "NeuroCartography: Scalable Automatic Visual Summarization of Concepts in Deep Neural Networks," in *IEEE Visualization Conference, (VIS)*, 2021.

[81] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[82] L. Liu *et al.*, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[83] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE international conference on acoustics, speech and signal processing*, IEEE, 2013, pp. 8599–8603.

[84] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, "Audio-visual speech recognition using deep learning," *Applied intelligence*, vol. 42, pp. 722–737, 2015.

[85] D. Ravi *et al.*, "Deep learning for health informatics," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 4–21, 2016.

[86] A. Rajkomar *et al.*, "Scalable and accurate deep learning with electronic health records," *NPJ digital medicine*, vol. 1, no. 1, pp. 1–10, 2018.

[87] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[88] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba, "Understanding the role of individual units in a deep neural network," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 071–30 078, 2020.

[89] N. Das *et al.*, "Bluff: Interactively deciphering adversarial attacks on deep neural networks," in *2020 IEEE Visualization Conference (VIS)*, IEEE, 2020, pp. 271–275.

[90] N. Das *et al.*, "Massif: Interactive interpretation of adversarial attacks on deep learning," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–7.

[91] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, "An overview of early vision in inceptionv1," *Distill*, vol. 5, no. 4, e00024–002, 2020.

[92] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1389–1397.

[93] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *BMVC*, 2014.

[94] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[95] R. Duggal, S. Freitas, C. Xiao, D. H. Chau, and J. Sun, "Rest: Robust and efficient neural networks for sleep monitoring in the wild," in *Proceedings of The Web Conference 2020*, 2020, pp. 1704–1714.

[96] R. Duggal, C. Xiao, R. Vuduc, D. H. Chau, and J. Sun, "Cup: Cluster pruning for compressing deep neural networks," in *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, 2021, pp. 5102–5106.

[97] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah, "Activation atlas," *Distill*, vol. 4, no. 3, e15, 2019.

[98] J. L. Long, N. Zhang, and T. Darrell, "Do convnets learn correspondence?" *Advances in neural information processing systems*, vol. 27, 2014.

[99] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *ICML DL Workshop*, 2015.

[100] C. Olah *et al.*, "The building blocks of interpretability," *Distill*, vol. 3, no. 3, e10, 2018.

[101] A. Z. Broder, "On the resemblance and containment of documents," in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, IEEE, 1997, pp. 21–29.

[102] O. Chum, J. Philbin, A. Zisserman, *et al.*, "Near duplicate image detection: Min-hash and tf-idf weighting.," in *Bmvc*, vol. 810, 2008, pp. 812–815.

[103] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: Scalable online collaborative filtering," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 271–280.

[104] A. Gionis, P. Indyk, R. Motwani, *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, 1999, pp. 518–529.

[105] A. Tamersoy, K. Roundy, and D. H. Chau, "Guilt by association: Large scale malware detection by mining file-relation graphs," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1524–1533.

[106] X. Hu, K. G. Shin, S. Bhatkar, and K. Griffin, "Mutantx-s: Scalable malware clustering based on static features," in *2013 {USENIX} Annual Technical Conference ({USENIX}{ATC} 13)*, 2013, pp. 187–198.

[107] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[108] Z. Yin and Y. Shen, "On the dimensionality of word embedding," *NeurIPS*, 2018.

[109] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[110] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ICLR Workshop*, 2013.

[111] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[112] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[113] J. Chang, S. Gerrish, C. Wang, J. Boyd-graber, and D. Blei, "Reading tea leaves: How humans interpret topic models," in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., vol. 22, Curran Associates, Inc., 2009.

[114] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recogn.*, vol. 30, no. 7, pp. 1145–1159, Jul. 1997.

[115] D. Cer *et al.*, "Universal sentence encoder," *arXiv preprint arXiv:1803.11175*, 2018.

[116] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, "Generalization in deep learning," in *Mathematical Aspects of Deep Learning*, Cambridge University Press, 2022.

[117] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[118] N. Cammarata, G. Goh, S. Carter, C. Voss, L. Schubert, and C. Olah, "Curve circuits," *Distill*, vol. 6, no. 1, e00024–006, 2021.

[119] 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*, 2016, pp. 770–778.

[120] Parliament and C. of the European Union, "General data protection regulation," 2016.

[121] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An approach to evaluating interpretability of machine learning," *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2018.

[122] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," in *IJCAI Workshop on Explainable AI*, 2017.

[123] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural nnetworks," *Digital Signal Processing*, 2017.

[124] F. Hohman, A. Head, R. Caruana, R. DeLine, and S. M. Drucker, "Gamut: A design probe to understand how data scientists understand machine learning models," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2019.

[125] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[126] Z. C. Lipton, "The mythos of model interpretability," *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[127] Y. Liu *et al.*, "Artificial intelligence-based breast cancer nodal metastasis detection.," *Archives of Ppathology & Laboratory Medicine*, 2018.

[128] D. F. Steiner *et al.*, "Impact of deep learning assistance on the histopathologic review of lymph nodes for metastatic breast cancer," *The American Journal of Surgical Pathology*, vol. 42, no. 12, pp. 1636–1646, 2018.

[129] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon, "Combining satellite imagery and machine learning to predict poverty," *Science*, vol. 353, no. 6301, pp. 790–794, 2016.

[130] N. Das *et al.*, "Shield: Fast, practical defense and vaccination for deep learning using jpeg compression," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 196–204.

[131] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3712–3722.

[132] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[133] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

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

[135] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4467–4477.

[136] C. Olah and S. Carter, "Research debt," *Distill*, 2017, https://distill.pub/2017/research-debt.

[137] M. Kahng, N. Thorat, D. H. P. Chau, F. B. Viégas, and M. Wattenberg, "Gan lab: Understanding complex deep generative models using interactive visual experimentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 310–320, 2019.

[138] A. Abdul, J. Vermeulen, D. Wang, B. Y. Lim, and M. Kankanhalli, "Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ACM, 2018, p. 582.

[139] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. 2016.

[140] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford InfoLab, Tech. Rep., 1999.

[141] 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.

[142] G. W. Furnas, *Generalized fisheye views*. Bell Communications Research. Morris Research and Engineering Center . . ., 1986, vol. 17.

[143] F. Van Ham and A. Perer, ""search, show context, expand on demand": Supporting large graph exploration with degree-of-interest," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 953–960, 2009.

[144] T. Crnovrsanin, I. Liao, Y. Wu, and K.-L. Ma, "Visual recommendations for network navigation," in *Computer Graphics Forum*, Wiley Online Library, vol. 30, 2011, pp. 1081–1090.

[145] S. Kairam, N. H. Riche, S. Drucker, R. Fernandez, and J. Heer, "Refinery: Visual exploration of large, heterogeneous networks through associative browsing," in *Computer Graphics Forum*, Wiley Online Library, vol. 34, 2015, pp. 301–310.

[146] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[147] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3429–3437.

[148] P.-J. Kindermans *et al.*, "Learning how to explain neural networks: Patternnet and patternattribution," *arXiv preprint arXiv:1705.05598*, 2017.

[149] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 3319–3328.

[150] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, no. 7, e0130140, 2015.

[151] H. Park *et al.*, "Skeletonvis: Interactive visualization for understanding adversarial attacks on human action recognition models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 16 094–16 096.

[152] A. Esteva *et al.*, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[153] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, 2019.

[154] G. Guo and N. Zhang, "A survey on deep learning based face recognition," *Computer Vision and Image Understanding*, vol. 189, p. 102 805, 2019.

[155] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19 143–19 165, 2019.

[156] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning for finance: Deep portfolios," *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 3–12, 2017.

[157] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[158] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, https://openreview.net/forum?id=rJzIBfZAb, Apr. 2018.

[159] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[160] F. Tramer, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *ICLR*, 2018.

[161] A. P. Norton and Y. Qi, "Adversarial-playground: A visualization suite showing how adversarial examples fool deep learning," in *Visualization for Cyber Security (VizSec), 2017 IEEE Symposium on*, IEEE, 2017, pp. 1–4.

[162] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2016, pp. 582–597.

[163] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[164] M. Kahng, N. Thorat, D. H. P. Chau, F. B. Viégas, and M. Wattenberg, "Gan lab: Understanding complex deep generative models using interactive visual experimentation," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 1–11, 2018.

[165] 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, 2018.

[166] W. Brendel and M. Bethge, "Approximating cnns with bag-of-local-features models works surprisingly well on imagenet," *ICLR*, 2019.

[167] A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick, "On the importance of single directions for generalization," *ICLR*, 2018.

[168] N. Das, S. Chaba, R. Wu, S. Gandhi, D. H. Chau, and X. Chu, "Goggles: Automatic image labeling with affinity coding," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1717–1732.

[169] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete, "Weighted graph comparison techniques for brain connectivity analysis," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2013, pp. 483–492.

[170] A. Saggese, N. Strisciuglio, M. Vento, and N. Petkov, "Learning skeleton representations for human action recognition," *Pattern Recognition Letters*, vol. 118, pp. 23–31, 2019.

[171] J. Liu, N. Akhtar, and A. Mian, "Adversarial attack on skeleton-based human action recognition," *Neural Networks And Learning Systems*, 2022.

[172] S. Freitas, S.-T. Chen, Z. J. Wang, and D. H. Chau, "Unmask: Adversarial detection and defense through robust feature alignment," *IEEE BigData*, 2020.

[173] H. Park, F. Hohman, and D. H. Chau, "Neuraldivergence: Exploring and understanding neural networks by comparing activation distributions," in *Poster, Pacific Visualization Symposium (PacificVis)*, IEEE, 2019.

[174] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*, 2019.

[175] H. Park *et al.*, "Concept evolution in deep learning training: A unified interpretation framework and discoveries," *CIKM*, 2023.

[176] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *International Conference on Machine Learning*, 2017.

[177] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," *arXiv preprint arXiv:1803.04765*, 2018.

[178] Q. Zhang, W. Wang, and S.-C. Zhu, "Examining cnn representations with respect to dataset bias," *AAAI Conference on Artificial Intelligence*, 2018.

[179] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, "This looks like that: Deep learning for interpretable image recognition," *Advances in neural information processing systems*, vol. 32, 2019.

[180] M. Abadi *et al.*, "{Tensorflow}: A system for {large-scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[181] Z. Zhou *et al.*, "Neuromapper: In-browser visualizer for neural network training," *IEEE VIS*, 2022.

[182] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," *Advances in neural information processing systems*, vol. 30, 2017.

[183] A. Ghorbani and J. Y. Zou, "Neuron shapley: Discovering the responsible neurons," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5922–5932, 2020.

[184] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2015.

[185] R. Fong and A. Vedaldi, "Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 8730–8738.

[186] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[187] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 976–11 986.

[188] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[189] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[190] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[191] A. Kolesnikov *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[192] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[193] J. V. Michalowicz, J. M. Nichols, and F. Bucholtz, *Handbook of differential entropy*. Crc Press, 2013.

[194] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 8093–8104.

[195] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations," *ICLR*, 2016.

[196] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *ICLR*, 2019.

[197] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[198] A. Ramesh *et al.*, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8821–8831.