SUMMIT: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations

Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng (Polo) Chau

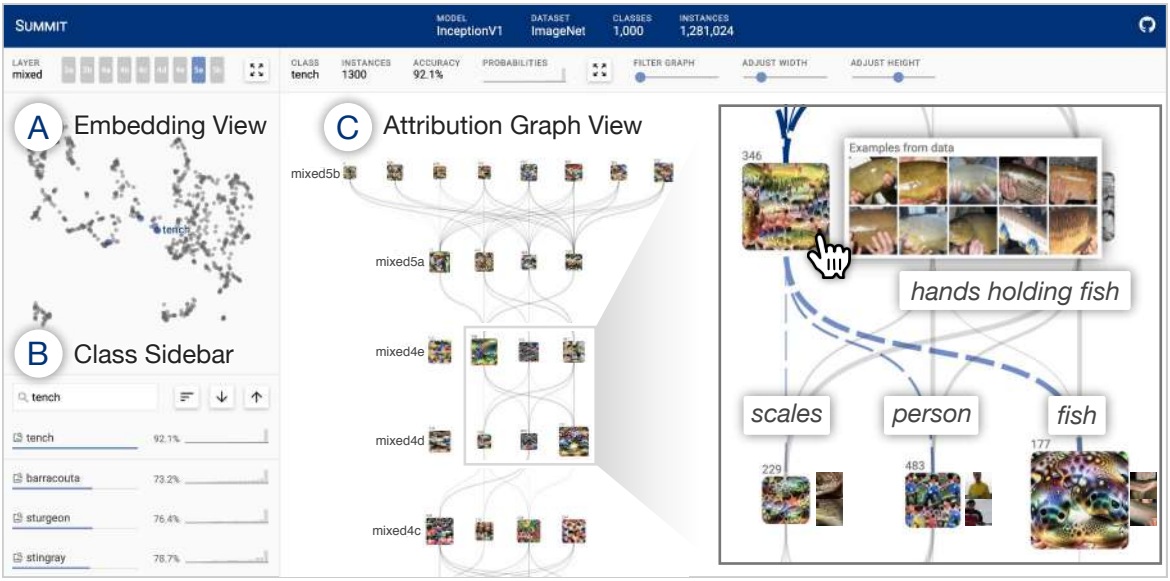

Fig. 1. With Summit, users can scalably summarize and interactively interpret deep neural networks by visualizing *what* features a network detects and *how* they are related. In this example, INCEPTIONV1 accurately classifies images of ***tench*** (yellow-brown fish). However, SUMMIT reveals surprising 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).

Abstract—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 from millions of weights that are optimized over millions of images, such explanations can easily miss a bigger picture. 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. 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 prevalent, large-scale image classifier's learned representations and informs future neural network architecture design. The SUMMIT visualization runs in modern web browsers and is open-sourced.

Index Terms—Deep learning interpretability, visual analytics, scalable summarization, attribution graph

1 INTRODUCTION

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 [44]. Definitions of interpretability center around *human understanding*, but they vary in the aspect of the model to be understood: its internals [16], operations [6], mapping of data [36], or representation [46]. Although recent work has begun to operationalize interpretability [21], a formal, agreed-upon definition remains open [10, 30].

Existing work on interpreting neural network predictions for im-

• *Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Chau are with Georgia Tech. E-mail: {fredhohman, haekyu, dcrobins, polo}@gatech.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

ages often focuses on explaining predictions for single images or neurons [40, 41, 49, 53]. 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 [46], since these representations are used in diverse tasks like detecting breast cancer [33, 54], predicting poverty from satellite imagery [23], defending against adversarial attacks [9], transfer learning [43, 59], and image style transfer [15]. 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 [24, 46]. As demonstrated in Fig. 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 [56]. SUMMIT leverages neural network feature visualization [11, 37, 38, 40, 50] and dataset examples to distill large, complex neural network models into compact, interactive graph visualizations (Sect. 7).
- **Two new scalable summarization techniques** for deep learning interpretability: (1) *activation aggregation* discovers important neurons (Sect. 6.1), and (2) *neuron-influence aggregation* identifies relationships among such neurons (Sect. 6.2). These techniques scale to large data, e.g., ImageNet ILSVRC 2012 with 1.2M images [47].
- **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. 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 (Sect. 6.3).
- **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 [25, 39, 52]. 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 (Sect. 8).

Broader impact for visualization in AI. 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 [1, 39].

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

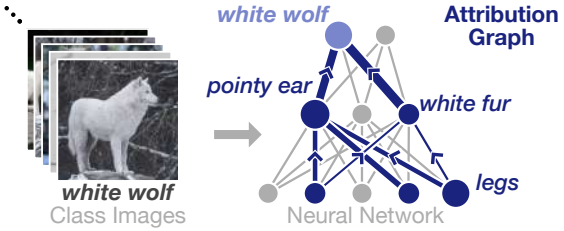

Fig. 2. A high-level illustration of how we take thousands of images for a given class, e.g., images from ***white wolf*** class, compute their top activations and attributions, and combine them to form an **attribution graph** that shows how lower-level features ("legs") contribute to higher-level ones ("white fur"), and ultimately the final outcome.

2 BACKGROUND FOR NEURAL NETWORK INTERPRETABILITY

Typically, a neural network is given an input data instance (e.g., an image) and computes transformations on this instance until ultimately producing a probability for prediction. Inside the network at each layer, each neuron (i.e., channel) detects a particular feature from the input. However, since deep learning models learn these features through training, research in interpretability investigates how to make sense of what specific features a network has detected. We provide an overview of existing activation-based methods for interpretability, a common approach to understand how neural networks operate internally that considers the magnitude of each detected feature inside hidden layers.

2.1 Understanding Neuron Activations

Neuron activations as features for interpretable explanations. There have been many approaches that use neuron activations as features for interpretable explanations of neural network decisions. TCAV vectorizes activations in each layer and uses the vectors in a binary classification task for determining an interpretable concept's relevance (e.g., striped pattern) in model's decision for a specific class (e.g., zebra) [27]. Network Dissection [4] and Net2Vec [12] propose methods to quantify interpretability by measuring alignment between filter activations and concepts. ActiVis visualizes activation patterns in an interactive table view, where the columns are neurons in a network and rows are data instances [24]. This table unifies instance-level and subset-level analysis, which enables users to explore inside neural networks and visually compare activation patterns across images, subsets, and classes.

Visualizing neurons with their activation. Instead of only considering the magnitude of activations, another technique called feature visualization algorithmically generates synthetic images that maximize a particular neuron [7, 11, 37, 38, 40, 50]. Since these feature visualizations optimize over a single neuron, users can begin to decipher what feature a single neuron may have learned. These techniques have provided strong evidence of how neural networks build their internal hierarchical representations [60]. Fig. 3 presents widely shared examples of how neural networks learn hierarchical features by showing neuron feature visualizations. It is commonly thought that neurons in lower layers in a network learn low-level features, such as edges and textures, while neurons in later layers learn more complicated parts and objects, like faces (Fig. 3). In our work, we crystallize this belief by leveraging feature visualization to identify what features a model has detected, and how they are related.

2.2 Towards Higher-level Deep Learning Interpretation

It is not uncommon for modern, state-of-the-art neural networks to contain hundreds of thousands of neurons; visualizing all of them is ineffective. To address this problem, several works have proposed to extract only "important" neurons for a model's predictions [7, 31, 41]. For example, Blocks, a visual analytics system, shows that class confusion patterns follow a hierarchical structure over the classes [5], and Activation Atlases, large-scale dimensionality reductions, show many

averaged activations [7]. Both visualizations reveal interesting properties of neural networks. However, they either (1) consider activations independent of their learned connections, (2) depend on randomized sampling techniques, or (3) are computationally expensive. SUMMIT addresses these issues by: (1) combining both activations and relationships between network layers, as only knowing the most important neurons is not sufficient for determining how a model made a prediction — the relationships between highly contributing neurons are key to understanding how learned features are combined inside a network; (2) leveraging entire datasets; and (3) integrating scalable techniques.

Since feature visualization has shown that neurons detect more complicated features towards a network's output, it is reasonable to hypothesize that feature construction is the collaborative combination of many different features from previous layers [4, 12, 48]. Our visualization community has started to investigate this hypothesis. For example, one of the earlier visual analytics approaches, CNNVis, derives neuron connections for model diagnosis and refinement, but did not scale to large datasets with many classes [32]. In the context of adversarial machine learning, AEVis uses backpropagation to identify where in a network the data paths of a benign and attacked instance diverge [31]. AEVis demonstrates its approach on single and small sets of images; it is unclear how the approach's integral approximation optimization techniques scale to large, entire datasets, such as ImageNet. Another example, Building Blocks, proposes to use matrix factorization to group sets of neurons together within a layer and derive "compatible" neuron groups across layers [41]; however, the work suggests uncertainty in the proposed formulation. Our work draws inspirations from the above important prior research in neural network visualization. Our method makes advances to scale to large million-image datasets, providing new ways to interpret entire classes (vs. single-image explanations) by aggregating activations and influences across the model.

2.3 Visual Analytics for Interpretability

To better facilitate interpretability, interactive visual analytics solutions have been proposed to help different user groups interpret models using a variety of interactive and visualization techniques [22]. Predictive visual analytics supports experts conducting performance analysis of machine learning models by visualizing distributions of predicted instances, computing feature importance, and directly inspecting model and instance errors to support debugging [2, 21, 34, 45, 58]. Interactive visualization for explaining models to non-experts using direct manipulation has also seen attention due to the pervasiveness of machine learning in modern society and general interest from the public [19, 25, 52].

3 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.

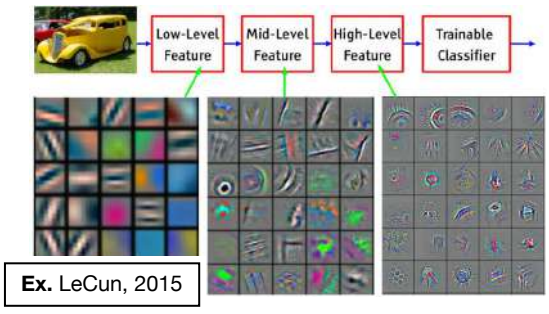

Fig. 3. A common, widely shared example illustrating how neural networks learn hierarchical feature representations. Our work crystallizes these illustrations by systematically building a graph representation that describe *what* features a model has learned and *how* they are related. We visualize features 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.

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 [40, 41, 49, 53]. 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 [17]. 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 [46], aiding decision making in disease diagnosis [33, 54], devising security protocols [9], and fixing under-performing models [24, 46].

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?

4 DESIGN GOALS

Based on the identified design challenges (Sect. 3), 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 (Sect. 6.1).

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 (Sect. 6.2).

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 (Sect. 6.3). 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 (Sect. 7). 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 (Sect. 7.3).

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 (Sect. 7). We also open-source our code to support reproducible research **(C5)**.

5 Model Choice and Background

In this work, we demonstrate our approach on INCEPTIONV1 [56], 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 [40, 41]. Although our work makes this model choice, our proposed summarization and visualization techniques can be applied to other neural network architectures in other domains.

6 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 (Sect. 6.3) for CNNs.

6.1 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 images 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 [60] (see Fig. 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, 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 $\sum^{m}_{j\in\text{sorted } Z'^l_i} Z'^l_{ij} \leq k_{M2}$, 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.

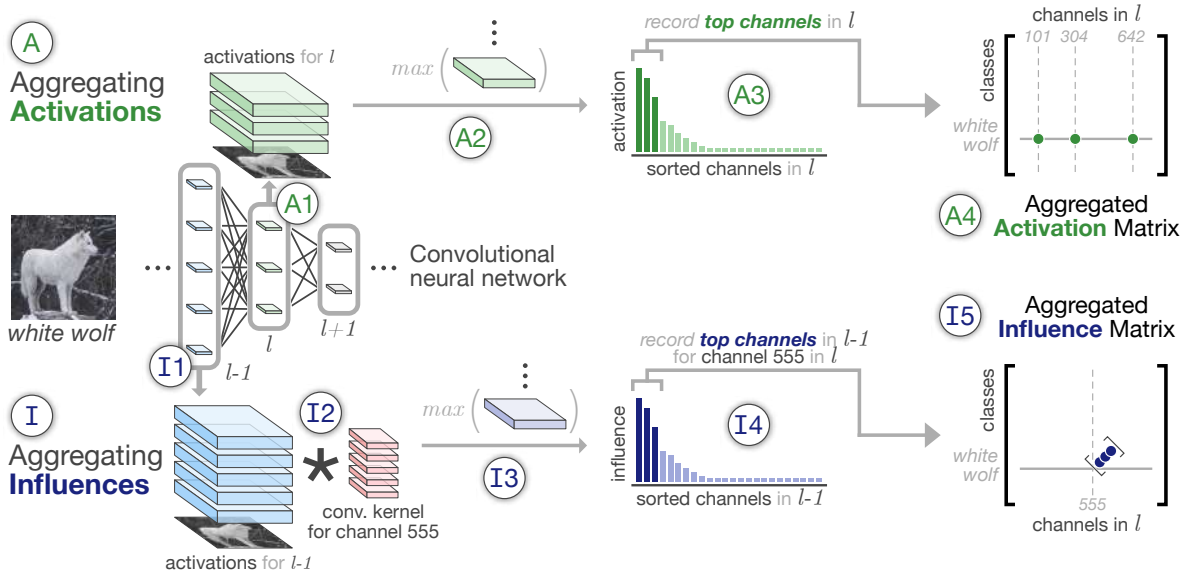

Fig. 4. Our approach for aggregating activations and influences for a layer l. **Aggregating Activations**: **(A1)** given activations at layer l, **(A2)** compute the max of each 2D channel, and **(A3)** record the top activated channels into an **(A4)** aggregated activation matrix, which tells us 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.

6.2 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:

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

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 Sect. 6.1, 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 (Sect. 6.1), 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 Sect. 6.2), or if L^l_{ij} is in the top $k_{M2}\%$ of largest values in $L^l_{:,j}$ (mirroring **Method 2** (Sect. 6.1).

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 Sect. 7.4.

6.3 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 [29, 42], 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 Sect. 6.1 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 [29, 42], Personalized PageRank runs in ~ 30 seconds for each class.

7 THE SUMMIT USER INTERFACE

From our design goals in Sect. 4 and our aggregation methodology in Sect. 6, we present SUMMIT, an interactive system for scalable summarization and interpretation for exploring entire learned classes in large-scale image classifier models (Fig. 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 Sect. 5, 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.

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

7.1 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. 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 [35]. 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. 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 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.

Fig. 5. Selectable network minimap animates the Embedding View.

7.2 Class Sidebar: Searching and Sorting Classes

Underneath the Embedding View sits the Class Sidebar (Fig. 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. 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.

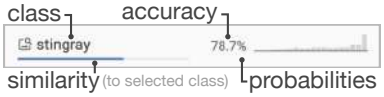

Fig. 6. Class Sidebar visual encoding.

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. 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. 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.

7.3 Attribution Graph View: Visual Class Summarization

The Attribution Graph View is the main view of SUMMIT (Fig. 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 Sect. 6.3 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 [32], AEVis [31], and Building Blocks [41], 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. 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 [40]; 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. 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 [40]. 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 [8, 14, 26, 57]. 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.

7.4 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 [40].

8 Neural Network Exploration Scenarios

8.1 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. 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. 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

[3] D3.js: `https://d3js.org/`

[4] Lucid: `https://github.com/tensorflow/lucid`

[5] `https://github.com/distillpub/post--feature-visualization`

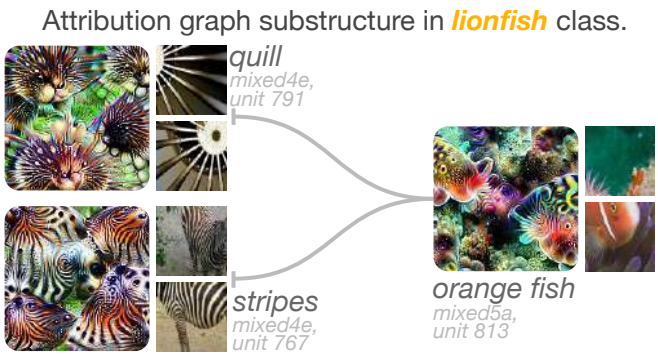

Fig. 7. An example substructure from the *lionfish* attribution graph that shows unexpected texture features, like "quills" and "stripes," influencing top activated channels for a final layer's "orange fish" feature (some *lion fish* are reddish-orange, and have white fin rays).

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. 7): there are 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.

8.2 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** classes, e.g., bison, wild boar, and ox. However, ***horsecart*** flips back at the final convolutional layer, returning to a **mechanical** association (Fig. 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. 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.

8.3 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 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. 9, we see in earlier layers (mixed4c) that both ***black bear*** and ***brown bear*** share

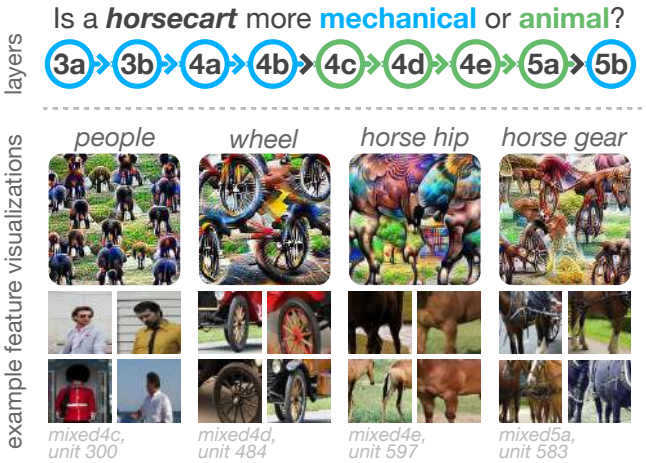

Fig. 8. Using SUMMIT we can find classes with mixed semantics that shift their primary associations throughout the network layers. For example, early in the network, ***horsecart*** is most similar to **mechanical** classes (e.g., harvester, thresher, snowplow), towards the middle it shifts to be nearer to **animal** classes (e.g., bison, wild boar, ox), but ultimately returns to have a stronger **mechanical** association at the network output.

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.

8.4 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. 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 produced high activation values but were incorrectly indicating important features in many classes. Although SUMMIT leverages recent feature visualization research [40] to visualize channels, it does not provide an automated way to measure the semantic quality of channels. We point readers to the appendix of [40] to explore this important future research direction.

8.5 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.

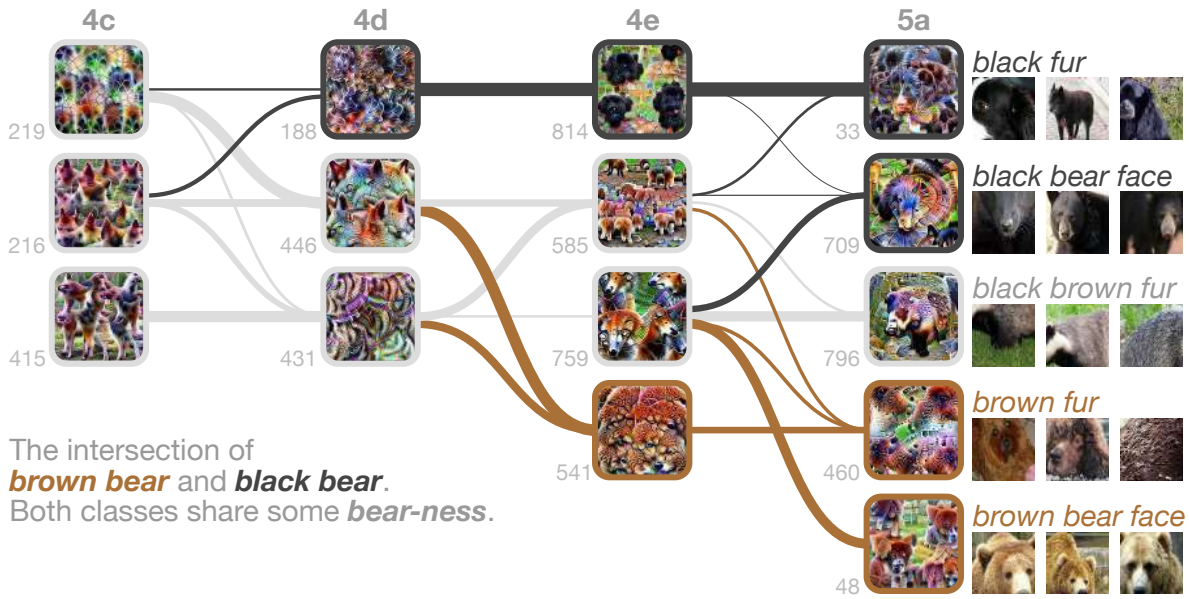

Fig. 9. With attribution graphs, we can compare classes throughout layers of a network. Here we compare two similar classes: ***black bear*** and ***brown bear***. From the intersection of their attribution graphs, we see both classes share features related to ***bear-ness***, but diverge towards the end of the network using fur color and face color as discriminable features. This feature discrimination aligns with how humans might classify bears.

Fig. 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.

9 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 Sect. 5, but an immediate avenue for future work explores generating attribution graphs on other CNN models. Simpler models like VGG [51] can be easily adapted with our approach, but more complex networks like ResNets [20] 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 [18], 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 [13, 28, 41, 49, 50, 55, 60]. 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 [3] 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.

10 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 [1, 39].

ACKNOWLEDGMENTS

We thank Nilaksh Das, the Georgia Tech Visualization Lab, and the anonymous reviewers for their support and constructive feedback. This work is supported by a NASA Space Technology Research Fellowship and NSF grants IIS-1563816, CNS-1704701, and TWC-1526254.

REFERENCES

[1] 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*, p. 582. ACM, 2018.

[2] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 337–346. ACM, 2015.

[3] 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*, 10(7):e0130140, 2015.

[4] 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*, pp. 6541–6549, 2017.

[5] A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do convolutional neural networks learn class hierarchy? *IEEE transactions on visualization and computer graphics*, 24(1):152–162, 2018.

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

[7] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah. Activation atlas. *Distill*, 4(3):e15, 2019.

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

[9] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau. 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*, pp. 196–204. ACM, 2018.

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

[11] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341:3, 2009.

[12] 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*, pp. 8730–8738, 2018.

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

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

[15] 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*, pp. 2414–2423, 2016.

[16] 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. *arXiv preprint arXiv:1806.00069*, 2018.

[17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. 2016.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[19] A. W. Harley. An interactive node-link visualization of convolutional neural networks. In *ISVC*, pp. 867–877, 2015.

[20] 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*, pp. 770–778, 2016.

[21] 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.

[22] 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*, 25(8):2674–2693, Aug 2019. doi: 10.1109/TVCG.2018.2843369

[23] 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*, 353(6301):790–794, 2016.

[24] 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*, 24(1):88–97, 2018.

[25] 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*, 25(1):310–320, 2019.

[26] 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*, vol. 34, pp. 301–310. Wiley Online Library, 2015.

[27] B. Kim, W. M., J. Gilmer, C. C., W. J., , F. Viegas, and R. Sayres. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). *ICML*, 2018.

[28] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

[29] A. N. Langville and C. D. Meyer. A survey of eigenvector methods for web information retrieval. *SIAM Review*, 47(1):135–161, 2005.

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

[31] M. Liu, S. Liu, H. Su, K. Cao, and J. Zhu. Analyzing the noise robustness of deep neural networks. *IEEE Conference on Visual Analytics Science and Technology*, 2018.

[32] 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*, 23(1):91–100, 2017.

[33] Y. Liu, T. Kohlberger, M. Norouzi, G. Dahl, J. Smith, A. Mohtashamian, N. Olson, L. Peng, J. Hipp, and M. Stumpe. Artificial intelligence-based breast cancer nodal metastasis detection. *Archives of Pathology & Laboratory Medicine*, 143(7):859–868, 2019.

[34] Y. Lu, R. Garcia, B. Hansen, M. Gleicher, and R. Maciejewski. The state-of-the-art in predictive visual analytics. In *Computer Graphics Forum*, vol. 36, pp. 539–562. Wiley Online Library, 2017.

[35] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, Feb. 2018.

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

[37] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog*, 2015.

[38] 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*, pp. 4467–4477, 2017.

[39] C. Olah and S. Carter. Research debt. *Distill*, 2017. https://distill.pub/2017/research-debt. doi: 10.23915/distill.00005

[40] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[41] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

[42] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[43] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[44] Parliament and C. of the European Union. General data protection regulation. 2016.

[45] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):61–70, 2017.

[46] 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*, pp. 1135–1144. ACM, 2016.

[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[48] R. R. Selvaraju, P. Chattopadhyay, M. Elhoseiny, T. Sharma, D. Batra, D. Parikh, and S. Lee. Choose your neuron: Incorporating domain knowledge through neuron-importance. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 526–541, 2018.

[49] 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*, pp. 618–626, 2017.

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

[51] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[52] D. Smilkov, S. Carter, D. Sculley, F. B. Viégas, and M. Wattenberg. Direct-manipulation visualization of deep networks. *ICML Workshop on Visualization for Deep Learning*, 2016.

[53] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: Removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[54] D. F. Steiner, R. MacDonald, Y. Liu, P. Truszkowski, J. D. Hipp, C. Gammage, F. Thng, L. Peng, and M. C. Stumpe. Impact of deep learning assistance on the histopathologic review of lymph nodes for metastatic breast cancer. *The American Journal of Surgical Pathology*, 42(12):1636–1646, 2018.

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

[56] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[57] 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*, 15(6):953–960, 2009.

[58] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mane, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE transactions on visualization and computer graphics*, 24(1):1–12, 2017.

[59] 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*, pp. 3712–3722, 2018.

[60] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pp. 818–833. Springer, 2014.