This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114858, IEEE Transactions on Visualization and Computer Graphics

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 (Polo) Chau

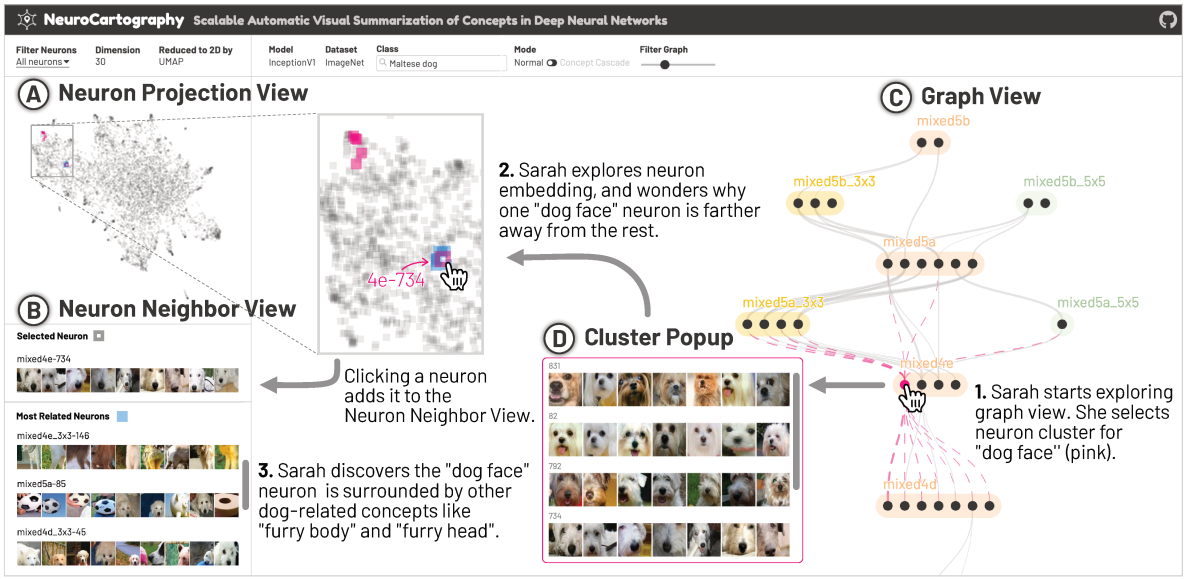

Fig. 1. NEUROCARTOGRAPHY scalably summarizes concepts learned by deep neural networks, by automatically discovering and visualizing groups of neurons that detect the same concepts. **1.** Our user Sarah starts exploring which neuron clusters (shown as circles) play an important role for InceptionV1 to 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.

Abstract— Existing research on making sense of deep neural networks often focuses on neuron-level interpretation, which may not adequately capture the bigger picture of how concepts are collectively encoded by multiple neurons. 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* 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). Key to our scalable techniques is the ability to efficiently compute all neuron pairs' relationships, in time linear to the number of neurons instead of quadratic time. NEUROCARTOGRAPHY scales to large data, such as the ImageNet dataset with 1.2M images. The system's tightly coordinated views integrate the scalable techniques to visualize the concepts and their relationships, projecting the concept associations to a 2D space in Neuron Projection View, and summarizing neuron clusters and their relationships in Graph View. 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 and isolated concepts. The NEUROCARTOGRAPHY visualization runs in modern browsers and is open-sourced.

Index Terms— Deep learning interpretability, visual analytics, scalable summarization, neuron clustering, neuron embedding

1 INTRODUCTION

Deep Neural Networks (DNNs) have demonstrated remarkable success in many applications, such as object detection [30, 60], speech recognition [13, 39], and data-driven health care [45, 46]. 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

The authors are at Georgia Institute of Technology. E-mail: {haekyu, nilakshdas, rahulduggal, apwright, oshaikh, fredhohman, polo}@gatech.edu. Fred Hohman is currently at Apple.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114858, IEEE Transactions on Visualization and Computer Graphics

to enable greater model interpretability [22, 57]. 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 [17, 28, 48, 49, 54]. 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 [1, 10, 11, 23, 42]. 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. 2 shows that the "dog face" concept is detected by multiple neurons in InceptionV1 model. Although it is a well-documented phenomenon that multiple neurons detect similar features [40] (especially in model pruning research [14, 21, 25, 56]), 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. 1), such as InceptionV1 [53]. 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 (Sect. 7).
- **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. 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 [47] (Sect. 6).
- **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 (Sect. 7.2.2).
- **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 (Sect. 8). 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 (Sect. 9).
- **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/.

2 RELATED WORK: NEURAL NETWORK INTERPRETABILITY

A Deep Neural Network (DNN) takes as input a data instance and outputs its class label. Transforming the input to the output may potentially involve billions of numerical computations, the scale of which is incomprehensible to humans. Recent research aims to summarize these calculations into abstract concepts, helping humans interpret how DNNs work. In this section, we provide an overview of existing research for neural network interpretability that motivate our work.

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

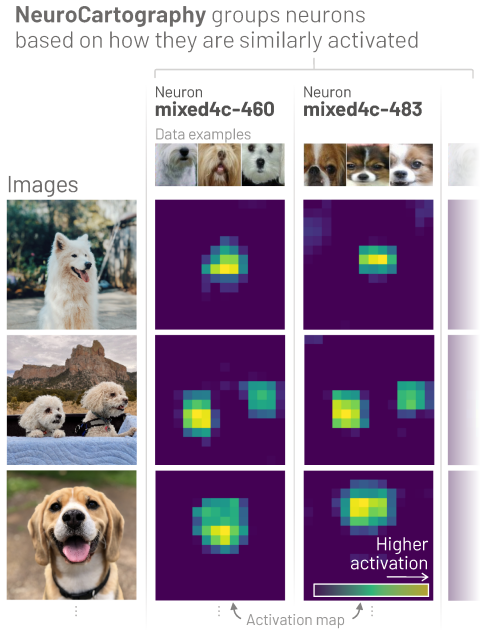

Fig. 2. To summarize the concepts learned by a DNN, NEUROCARTOGRAPHY groups neurons based on how similarly they are activated, e.g., by the "dog face" concept. Here, neurons 460 and 483 in layer mixed4c of InceptionV1 model are similarly activated by the "dog face" concept, and are grouped in the same cluster by our approach.

2.1 Interpretation at Input-, Layer-, and Neuron-Level

A common approach to interpret how DNNs operate internally is to study **features** detected by the models. Saliency Maps [50] and Grad-CAM [48] explain a DNN's prediction at the *input-level*, by finding contributing features from the input images. TCAV [28] explains a feature's importance for a class at the *layer-level*, by vectorizing activations in a layer and measuring the vector's sensitivity to the classes. A growing number of techniques [1, 16, 23, 38] aim to interpret such features at the *neuron-level*. Given a predefined segmented image that highlights regions for human-interpretable concepts, Net2vec [16] represents these concepts as the linear combination of thresholded activation maps from activated neurons. These techniques quantitatively determine how each neuron contributes to the concept image. Feature-visualization techniques [37, 38, 42], on the other hand, algorithmically generate synthetic images that maximally activate a particular neuron, thus visualizing what features are preferred by each neuron. Other methods [42] also generate examples cropped from real images in the dataset that highly activate a specific neuron. In NEUROCARTOGRAPHY, we use dataset examples to represent features detected by each neuron.

Some research focuses on interpreting the **relationships among features** in the form of embedding vectors [5, 26, 38]. Karpathy's t-SNE feature representation [26] visualizes similarity among features at the *input-level*, by projecting each image onto 2D space using the image's DNN features (from a full-connected layer). Nguyen et al. [38] represent multifaceted features of a neuron at the *input-level*. For example, it embeds onto 2D space all bell pepper images that activate a neuron detecting bell peppers, revealing the neuron's facets (e.g., red, green, yellow peppers). Activation Atlas [5] computes embeddings at the *layer-level*, where each embedded point represents a linear combination of all neurons' activation strengths in a layer, based on spatial patches extracted from those neurons' activation maps. As the above techniques generate embeddings at abstraction levels different from ours, it is unclear how they may directly map each point in the embedding space back to specific neurons. In other words, existing

approaches cannot pinpoint which neurons are responsible for detecting specific features, which NEUROCARTOGRAPHY supports (Sect. 6.2).

2.2 Semantic Similarity of Neurons

Recent research [15, 21, 25, 40, 56] suggests that neurons tend to detect similar features. Fig. 2 illustrates this claim by highlighting how different neurons detect the same feature of a dog's nose. Olah et al. [40] highlights many examples of similar neurons in InceptionV1 and visualizes which concepts are detected by such neurons; however, the examples are manually curated by the authors. Identifying neurons that discover similar concepts also has practical benefits: in the neural network compression community, several methods [14, 15, 21, 25, 56] leverage potential neuron redundancies to generate compressed models while maintaining prediction accuracy. Even though these methods can measure neurons' similarity, there is limited work in interpreting their *semantic* similarity. CNNPruner [29], an interface for pruning neurons, helps users understand the role of neurons through the filter visualization technique [52]. However, it describes its pruning approach mainly based on metrics that measure instability and sensitivity of neurons, ignoring groupings of semantically similar neurons. We draw inspiration from the above important prior research to automatically find groups of similar neurons and interpret semantic similarities among them.

2.3 Connection among Neurons

A key role in neural networks is played by neurons. Neurons are responsible for receiving and transmitting activation signals. However, they do not work in isolation. For a neuron to detect a feature, it requires an orchestrated interaction among many neurons across different layers. Recent research [10, 11, 23, 41] visually explains how higher level concepts can be constructed by neural connections. In the context of adversarial attack, some methods [10, 11, 31] identify *where* in a network the activation pathways of a benign and attacked input instance diverge, and how those diverging activations arrive at an incorrect prediction through connections among neurons. Inspired by these techniques, we summarize and visualize how neuron groups interact through connections among them, providing a new way to interpret concept cascading across layers.

3 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 [23, 37, 42]. 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 [15, 21, 25, 56]. 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 [37, 42]. 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")?

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 [5, 23]. 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?

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 [31, 41]. Recent research has proposed approaches to help users interpret how features may be connected [11, 23], 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"?

4 DESIGN GOALS

We distill the design challenges identified in Sect. 3 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 [32, 59]. 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 (Sect. 6.1) and (2) visualizing the concepts that are detected by such neuron groups (Sect. 7.2).

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 (Sect. 6.2). 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) (Sect. 7.1).

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 (Sect. 6). 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 (Sect. 6.1). 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 (Sect. 6.2).

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 (Sect. 7.2.2).

5 Model Choice and Background

In this work, we demonstrate our approach using InceptionV1 [35], 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 [42,43]. 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.

6 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 Sect. 6.1, describe neuron embeddings in Sect. 6.2, and detail how we filter concepts that are important for the prediction of each class in Sect. 6.3.

6.1 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. 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.

6.1.1 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. (1)

$$SimTopImgs(i,j) = \frac{|X_i \cap X_j|}{|X_i \cup X_j|} \tag{1}$$

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 [3, 8, 9, 18, 55].

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. (1) involves. Min-Hashing [3] 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, $h_{\min}(S) = \min_{s \in S}(h(s))$. 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, $Pr[h_{\min}(S_1) = h_{\min}(S_2)] =$ 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 [18, 24, 44]. 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 [44].

6.1.2 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. (2), where $\wedge$ and $\vee$ are element-wise `and` and `or` operation respectively, and $\texttt{numTrue}(\cdot)$ returns the number of true values in the input matrix. If $\texttt{numTrue}(Q_i(\mathbf{x}) \vee Q_j(\mathbf{x})) = 0$, the similarity between i and j is defined as 0.

$$SimActMap(i,j) = \frac{\texttt{numTrue}(Q_i(\mathbf{x}) \wedge Q_j(\mathbf{x}))}{\texttt{numTrue}(Q_i(\mathbf{x}) \vee Q_j(\mathbf{x}))} \tag{2}$$

The similarity $SimActMap(i,j)$ in Eq. (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.

6.1.3 Hyperparameter Selections for Neuron Clustering

Our neuron clustering approach uses a few hyperparameters: 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.

6.2 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. (Sect. 6.2.1)
- **Step 2.** Reduce the dimensions of the learned vector representation to 2D for visualization (Sect. 6.2.2), which we will describe in Sect. 7.1.

The decision to adopt a two-step approach to first generate higher-dimensional vector representations for neurons (Step 1) was motivated by prior work [19, 34, 58], where abstract concepts are better captured by higher-dimensional representations, which opens up the possibilities for supporting interpretation tasks at higher fidelity.

6.2.1 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), 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., $\sigma(x) = 1/(1+e^{-x})$) .

$$J = \sum_{(i,j) \in D} -\log(\sigma(V_i \cdot V_j)) \tag{3}$$

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 [34] 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 (Sect. 6.1.1). 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. (4), where M is the size of negative sampling for a pair of related neurons (i,j).

$$J = \sum_{(i,j) \in D} \Bigg(-\log(\sigma(V_i \cdot V_j)) + \sum_{m=1}^{M} \big(-\log(1-\sigma(V_i \cdot V_m)) - \log(1-\sigma(V_j \cdot V_m)) \big) \Bigg) \tag{4}$$

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. (5) and (6).

$$\frac{\partial J}{\partial V_i} = -\big(1-\sigma(V_i \cdot V_j)\big)V_j + \sum_{m=1}^{M} \sigma(V_i \cdot V_m)V_m \tag{5}$$

$$\frac{\partial J}{\partial V_j} = -\big(1-\sigma(V_i \cdot V_j)\big)V_i + \sum_{m=1}^{M} \sigma(V_j \cdot V_m)V_m \tag{6}$$

We update the embedding by gradient descent as in Eq. (7), where γ is the learning rate.

$$V_i \leftarrow V_i - \gamma \frac{\partial J}{\partial V_i}, \quad V_j \leftarrow V_j - \gamma \frac{\partial J}{\partial V_j} \tag{7}$$

6.2.2 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 [33].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114858, IEEE Transactions on Visualization and Computer Graphics

6.2.3 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.

6.3 Filtering Each Class's Important Model Substructures

6.3.1 Important Neurons and Neuron Groups

Our goal is to summarize important model substructures (i.e., neurons or neuron groups) that contribute to a model's class prediction. To do so, we follow an approach similar to [23], 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 [23].

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.

6.3.2 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 [23] 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 [23] for a more in-depth treatment of this approach. Our implementation deviates from [23] 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 $G2$.

7 User Interface

Based on our design goals in Sect. 4 and our neural network summarization techniques in Sect. 6, we present NeuroCartography, an interactive system that summarizes concepts detected by neuron clusters (Fig. 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 (Sect. 7.1), and (3) Graph View visualizes concept relations (Sect. 7.2).

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

The Neuron Projection View (Fig. 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 Sect. 6.2 onto a 2D space.

Fig. 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).

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). 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). 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.

7.2 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. 1C). Graph View provides two modes through toggle button in the header: (1) class exploration mode (Sect. 7.2.1) to visualize concepts important for a user-selected class and (2) concept cascade mode (Sect. 7.2.2) to selectively activate a concept and examine its cascade effect.

7.2.1 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 (Sect. 6.3). The nodes (circles) are neuron clusters or individual neurons within the same layers, displayed in the order of their important scores computed in Sect. 6.3.1. The edges represent influence among the nodes, where edge weights are computed in Sect. 6.3.2. 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. 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114858, IEEE Transactions on Visualization and Computer Graphics

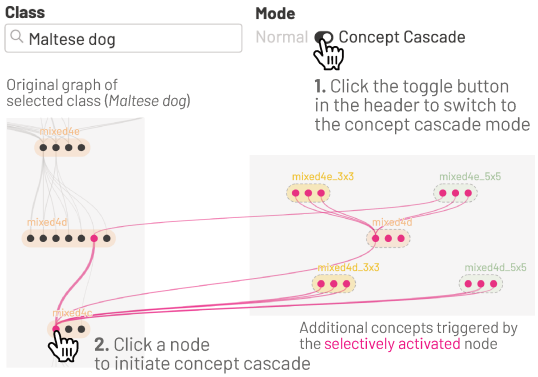

Fig. 4. Clicking a concept cluster node activates the concept, causing the neuron cluster to induce a **concept cascade** that triggers higher-level concepts across subsequent layers in the model. **Left:** in the concept cascade, some neuron clusters are strongly contributing to current class's prediction and are shown in the class's graph summary. **Right:** Concepts less related are shown on the right hand side.

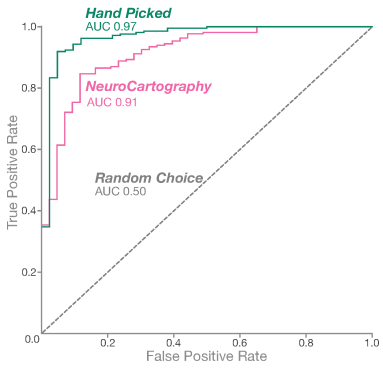

Fig. 5. ROC Curve for user estimations of cluster inclusion. Both hand picked and NEUROCARTOGRAPHY generated clusters perform well overall, implying that the clusters generated are interpretable enough to be consistently recognised by different users.

Projection View. Users can filter neurons in the Neuron Projection View to focus on pinned nodes using the dropdown menu in the header.

7.2.2 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. 4, at 1). Then, users can click a concept cluster node to select it and manually activate the selected concept cluster (Fig. 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. 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. 4, right).

7.3 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.

8 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 [12], basing our experimental design on similar work in image [17] and language [7] based cluster interpretability studies. For the experiment each user was presented a series of tasks like the task shown in Fig. 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 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 [7, 17]. 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. 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.

8.1 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114858, IEEE Transactions on Visualization and Computer Graphics

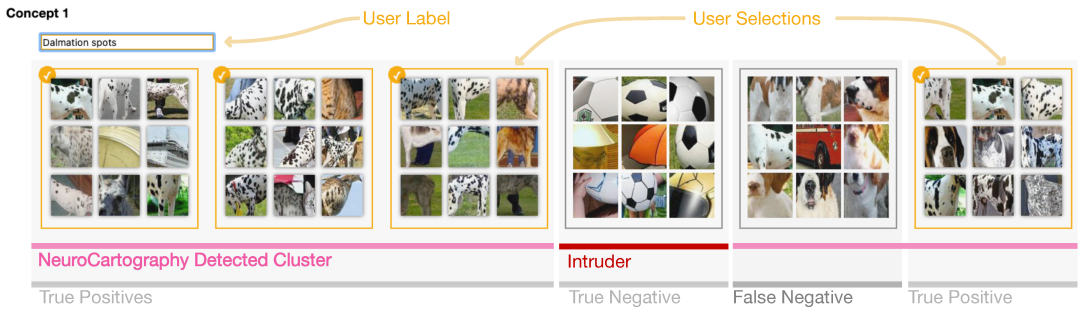

Fig. 6. Example question from MTurk evaluation. Users were presented with six neuron patches and asked to determine if there is a coherent cluster and if so provide a short label. In this example of a NEUROCARTOGRAPHY generated cluster we can see which neuron is the out of cluster intruder, which neurons 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.

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 [2].

The score used to define the ROC in Fig. 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. 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.

8.2 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) [6]. 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 [17], 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).

9 USAGE SCENARIOS

9.1 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 [27,36]. 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 [23], 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. 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. 8). Existing work [4] 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. 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.

9.2 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. 9, right). Inspecting such isolated neurons reveals that they are detecting the "watermark" concept (see

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114858, IEEE Transactions on Visualization and Computer Graphics

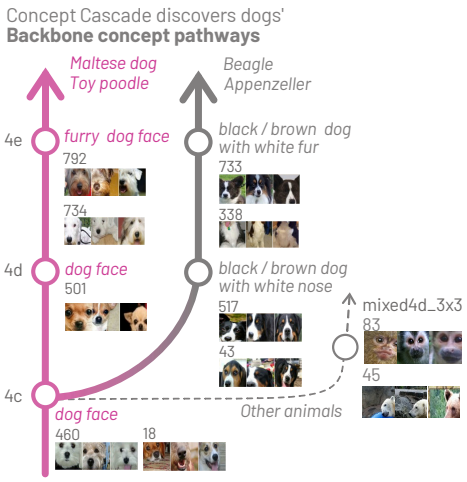

Fig. 7. Through Concept Cascade, users manually activate the "dog face" concept to discover its related concepts in subsequent layers, not only for the current "Maltese dog" class but also for other breeds of dogs. Concept Cascade helps users visualize how concepts may evolve over the network, such as from the more generic "dog face" concept to the more specific "furry dog face" concept in later layers. Concepts are manually labeled.

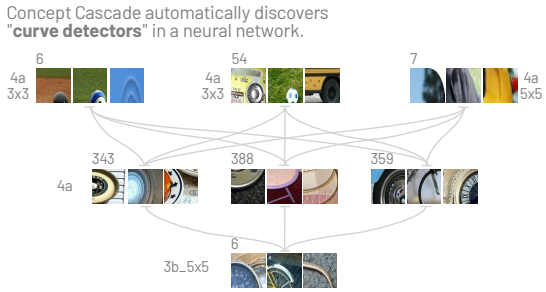

Fig. 8. Concept Cascade automatically discovers neurons that detects curve of specific orientations, which have been manually found in [4].

example from mixed5b-337 layer at Fig. 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. 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).

10 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.

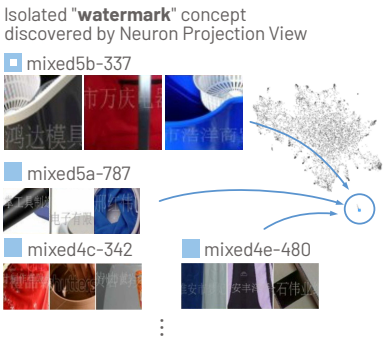

Fig. 9. NEUROCARTOGRAPHY reveals the interesting phenomenon about the isolated "watermark" concept (example image at top-left), that watermarks are not specific to any image features (i.e., can appear on almost any kinds of images), thus watermark neurons are placed far away from all other neurons due to relatively low co-activations (see Neuron Projection View on the right).

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., [1, 16, 23, 38]) 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, [41]. For example, in InceptionV1, at least poly-semantic neuron that responds to cat faces, fronts of cars, and cat legs [41]. NEUROCARTOGRAPHY cannot "split" this neuron into 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 [15, 21, 25, 56] 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 Sect. 5; we are working to extend support to other CNN models. Our approach can easily be adapted to simpler models (e.g., VGG [51]). For more complex networks (e.g., ResNets [20], 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).

Acknowledgments

We thank Hannah Kim, the Georgia Tech Visualization Lab, and the anonymous reviewers for their support and constructive feedback. This work was supported in part by DARPA (HR00112030001), NSF grants IIS-1563816, CNS-1704701, and gifts from Intel, NVIDIA, Google.

REFERENCES

[1] 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*, 117(48):30071–30078, 2020.

[2] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, July 1997. doi: 10.1016/S0031-3203(96)00142-2

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

[4] N. Cammarata, G. Goh, S. Carter, C. Voss, L. Schubert, and C. Olah. Curve circuits. *Distill*, 6(1):e00024–006, 2021.

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

[6] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Céspedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

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

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

[9] 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*, pp. 271–280, 2007.

[10] N. Das, H. Park, Z. J. Wang, F. Hohman, R. Firstman, E. Rogers, and D. H. Chau. Massif: Interactive interpretation of adversarial attacks on deep learning. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–7, 2020.

[11] N. Das, H. Park, Z. J. Wang, F. Hohman, R. Firstman, E. Rogers, D. Horng, et al. Bluff: Interactively deciphering adversarial attacks on deep neural networks. *arXiv preprint arXiv:2009.02608*, 2020.

[12] 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*, pp. 248–255. Ieee, 2009.

[13] 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*, pp. 8599–8603. IEEE, 2013.

[14] 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*, pp. 1704–1714, 2020.

[15] R. Duggal, C. Xiao, R. Vuduc, and J. Sun. Cup: Cluster pruning for compressing deep neural networks. *arXiv preprint arXiv:1911.08630*, 2019.

[16] 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.

[17] A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, pp. 9277–9286, 2019.

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

[19] 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*, pp. 855–864, 2016.

[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] 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*, pp. 1389–1397, 2017.

[22] F. Hohman, H. Park, C. Robinson, and D. H. Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2020.

[23] F. Hohman, H. Park, C. Robinson, and D. H. P. Chau. S ummit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE transactions on visualization and computer graphics*, 26(1):1096–1106, 2019.

[24] 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)*, pp. 187–198, 2013.

[25] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[26] A. Karpathy. t-sne visualization of cnn codes. *https://cs.stanford.edu/people/karpathy/cnnembed/*.

[27] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[28] 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*, pp. 2668–2677. PMLR, 2018.

[29] G. Li, J. Wang, H.-W. Shen, K. Chen, G. Shan, and Z. Lu. Cnnpruner: Pruning convolutional neural networks with visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[30] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[31] 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)*, pp. 60–71. IEEE, 2018.

[32] J. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? *arXiv preprint arXiv:1411.1091*, 2014.

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

[34] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

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

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

[37] 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 (NeurIPS)*, 2016.

[38] A. Nguyen, J. Yosinski, and J. Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *Workshop on Visualization for Deep Learning, International Conference on Machine Learning (ICML)*, 2016.

[39] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata. Audio-visual speech recognition using deep learning. *Applied Intelligence*, 42(4):722–737, 2015.

[40] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. An overview of early vision in inceptionv1. *Distill*, 5(4):e00024–002, 2020.

[41] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

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

[43] 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.

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

[45] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):1–10, 2018.

[46] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang. Deep learning for health informatics. *IEEE journal of biomedical and health informatics*, 21(1):4–21, 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, 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.

[49] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[50] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

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

[52] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *ICLR Workshop*, 2014.

[53] 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.

[54] 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*, pp. 2818–2826, 2016.

[55] 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*, pp. 1524–1533, 2014.

[56] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665*, 2016.

[57] A. P. Wright, Z. J. Wang, H. Park, G. Guo, F. Sperrle, M. El-Assady, A. Endert, D. Keim, and D. H. Chau. A comparative analysis of industry human-ai interaction guidelines. 2020.

[58] Z. Yin and Y. Shen. On the dimensionality of word embedding. *arXiv preprint arXiv:1812.04224*, 2018.

[59] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[60] 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*, 30(11):3212–3232, 2019.