Visualizing Deep Neural Nets with UMAP Tour

Anonymous Authors[1]

Abstract

Neural networks should be interpretable to humans. In particular, there is a growing interest in what concepts a particular layer has learned in training. In this work, a tool, UMAP Tour, is built and used to visually inspect and compare internal behavior of real-world models with instance-level details. The method used in the visualization also implies a new similarity measure between neural network layers. Using the visual tool and the similarity measure, we find concepts learned in state-of-the-art models and dissimilarities between them, such as GoogLeNet and ResNet.

1. Introduction

Modern neural networks outperform humans in a number of learning tasks (Szegedy et al., 2015; Sun et al., 2014; Lan et al., 2020). However, due to their over-parameterized, non-linear behavior, it is often difficult to reason about how they approach these tasks. Often, we try to understand deep neural networks via their internal representations (Goldfeld et al., 2018). To compare these internal representations, different notions of similarity have been proposed (Raghu et al., 2017; Morcos et al., 2018; Kornblith et al., 2019). In particular, Kornblith et al. used centered kernel alignment (CKA) to measure the similarity between representations in layers of neural network models (Kornblith et al., 2019). Here, we follow their exposition. Given a set of n input instances to the learning task, let $X \in \mathbb{R}^{n \times p_1}$ be a matrix of neuron activations for the n examples in some layer of a neural network. Let $Y \in \mathbb{R}^{n \times p_2}$ be the neuron activation matrix defined in the same fashion for another layer, which can come from the same or a different network. For each pair of activations X and Y in two layers, a *similarity index* defines a scalar function from the pair (X, Y) to $s(X, Y) \in \mathbb{R}$, where a higher score indicates a higher similarity between the two layer representations.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Compared to other existing similarity indices, CKA (Kornblith et al., 2019) satisfies three favorable properties: CKA is invariant to orthogonal transformations and isotropic scaling, but *not* invariant to general invertible transformation. In addition, Kornblith et al. showed that CKA captures common features in neural network layers, such as similarities between skip connections, as well as correspondences between layers in different neural network architectures.

Similarity indices such as CKA give a reliable summary of behavioral differences in different layers. However, a single score can be insufficient to describe instance-level differences between the two layers, and instance-level explanations are often features of effective methods (Kim et al., 2018). Here, we propose a method to measure and use layer similarity through orthogonal Procrustes analysis (Gower et al., 2004). The orthogonal Procrustes problem has a close relation to CKA. By design, it naturally satisfies the same set of invariances that CKA satisfies. In addition, it gives us a natural way to *align* representations between two layers. Through this alignment, we can provide *instance-level* visual comparisons between layer representations through animations or static side-by-side views. We will demonstrate through examples that Procrustes analysis gives a reliable similarity measure as CKA does, and show use cases of utilizing such similarity index to visually compare neural network models in our interactive visual system.

Our contributions can be summarized as follows:

- we propose a similarity index between layers which satisfies all of the three favorable invariance properties by (Kornblith et al., 2019);
- we use this index to derive a reliable *alignment* between data representations of different layers of neural network architectures;
- we derive a direct manipulation method to visually and interactively compare internal behavior of different layers through their behaviors on input instances;
- we use our system to identify important events and patterns a) when passing training data through the network layers; b) when passing external data through the network layers; and c) when comparing two neural network architectures.

In the following sections, we will: introduce necessary background and related work in Section 2; describe our method in Section 3; show and discuss experimental results in Section 4.

2. Background and Related Work

2.1. Dimensionality Reduction

Due to its over-parametrized design, neural network processes and represents the input signal in very high dimensional space. This high dimensional representation can be difficult to understand or visualize. Fortunately, this representation is often overcomplicated, as most of its signal embeds in a lower dimensional manifold. Many approaches has been used to reduce high dimensional data into low dimensional representation (Wold et al., 1987; Borg & Groenen, 2005; Tenenbaum et al., 2000; Roweis & Saul, 2000; Van der Maaten & Hinton, 2008; McInnes et al., 2018). In particular, Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) can reliably preserve the global structure of high dimensional data in the low dimensional embedding.

2.1.1. UMAP

UMAP works by finding an embedding that resembles the (weighted) neighbor graph of data points in the high dimensional space. In practice, UMAP scales well with the number of data points (McInnes, 2018) and has little computational overhead on the embedding dimensionality, which make it ideal for projecting data to more than 2 or 3 dimensions. Projecting data to more than 3 dimensions can be useful in many contexts. For instance, one can extract low dimensional features from the original data and use the learned features for any downstream machine learning tasks. We use multi-dimensional UMAP to extract important features from the very high dimensional neuron activations, hoping that more than three dimensional embeddings can capture more important structures in the data. First, we will validate this idea by quantifying UMAP's efficacy when the embeddings have more than 3 dimensions.

As we have described, UMAP works by preserving neighbor graph. UMAP achieves this goal by finding an embedding that minimizes the binary cross entropy (BCE) loss between the true neighborhood graph in the data space and the *induced* neighborhood graph in the embedding space. A lower loss indicates a better embedding. Having more dimensions in the embedding space naturally gives UMAP more room to optimize the layout and therefore yields lower loss value. Here we evaluate the impact of embedding dimensions on neuron activation data. In Figure 1, we plot the final loss of UMAP as we vary the embedding dimensions. When applying UMAP on the activation of 15-Inception layer of

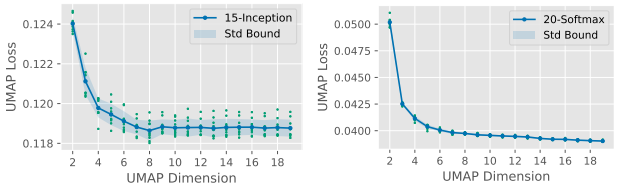

Figure 1: Analysis of UMAP embedding dimensions. We picked two layers of GoogLeNet, compute the UMAP embeddings of ImageNet's validation set, with different embedding dimensions (each with 10 random initial states for UMAP). **Left:** In the '15-Inception' layer (40768-D), the loss decreases until 8 dimensions and plateaus afterward. **Right:** In 20-Softmax (1000-D), loss keeps decreasing till the largest number of dimensions considered is reached.

a well-trained GoogLeNet model (Figure 1, left), we see a large gain when we increase embedding dimension from 2D to 4D, followed by smaller improvement from 4D to 8D, and a plateau afterward. On the final softmax layer (Figure 1, right), we see the loss continues to improve until as far as 19 dimensions. In the supplementary material we report the same analysis on other layers of GoogLeNet. This empirical analysis of UMAP dimensionality suggests a potential benefit of embedding data to more than 3 dimensions. In theory, one should embed data into a space with dimensionality no less than the intrinsic dimensionality of the data manifold, as n-manifolds can not be homeomorphically embedded in less than n dimensions. In practice, however, it is not straightforward to estimate the intrinsic dimensionality of data manifolds and choose the optimal embedding dimension accordingly. Throughout the experiments and visualizations in this work, we chose 15 dimensional UMAP in order to balance the preciseness of structure in the embedding and the runtime performance of the visualization.

2.2. Visualization

People have been visualizing neural networks via different angles, including saliency maps (Selvaraju et al., 2017; Zeiler & Fergus, 2014; Bach et al., 2015; Ancona et al., 2019), feature visualizations (Olah et al., 2017), embedding methods (Karpathy, 2012; Rauber et al., 2016; Li et al., 2020) or combinations of some approaches (Olah et al., 2018; Carter et al., 2019). While most embedding-based visualizations project the high-dimensional data into 2 or 3 dimensions, Li et al. linearly project the internal representation onto more than 3 dimensions (using PCA) and used the Grand Tour to visualize the neuron activations in small neural networks (Li et al., 2020). We take a similar visual approach to Li et al., but extend their method in two aspects: scalability and alignment mechanism. Instead of linearly project neuron activations, we non-linearly project them via UMAP. Our method is more flexible and suitable

for larger and real-world models. Besides, Li et al. align *consecutive* layer representations using the singular value decomposition (SVD) of fully connected layers. In contrast, we use a more general method, namely orthogonal Procrustes (Section 2.3.2), to align *any* pair of layers. Our method works even when the two layers come from different neural network architectures. Due to this, our method is used to compare neural architectures (Section 4).

2.2.1. The Grand Tour

The Grand Tour (Asimov, 1985) is a general method to visualize high dimensional data. It works by rotating the multi-dimensional data points and projecting it to 2D for display. It is a multi-dimensional analogy to 'filming a spinning 3D object and display it on a 2D screen'. People make sense of high dimensional data through this animated 2D scatter plot which is rendered from a smooth sequence of 2D projections. Formally, the Grand Tour designs a smooth sequence of orthogonal matrices $GT_t \in \mathbb{R}^{p\times p}$ (e.g. using torus method (Asimov, 1985)) which is parametrized by time step t. It then uses this sequence to project p-dimensional data points $X \in \mathbb{R}^{n\times p}$ and we visualize the first 2 components of the projection. In Grand Tour, we visualize smoothly animated data points $\{(x_i(t), y_i(t)), i = 1 \dots n\}$ in a 2D scatter plot, where each data point $X_i \in \mathbb{R}^{1\times p}$ is processed by Grand Tour rotation GT_t followed by an orthogonal projection $P_2 : (x_1, x_2, \dots, x_p) \mapsto (x_1, x_2)$ onto the xy-plane:

$$(x_i(t), y_i(t)) = X_i \cdot GT_t \cdot P_2 \tag{1}$$

Several extensions of the Grand Tour exist in the literature (Cook et al., 1995; Cook & Buja, 1997; Cook et al., 2008). In particular, (Cook & Buja, 1997) provides *manual controls* over the projection through tuning the contribution of each variable. Instead of controlling the contribution of variables in the projection (Cook & Buja, 1997), we take a more intuitive approach proposed by (Li et al., 2020), which directly manipulate group of data points to navigate the Grand Tour view. Our interface allows the viewer to directly drag a subset of data points to any desired location. This behavior is achieved by deriving an optimal Grand Tour projection that fulfill viewer's specification. We will explain in more detail in Section 3.2.1.

2.3. Similarity Measures and Alignments

Previous studies have proposed different notions of similarities between neural network representations (Raghu et al., 2017; Morcos et al., 2018; Kornblith et al., 2019). In particular, Kornblith et al. observed that centered kernel alignment (CKA) has three desirable invariance properties that others do not always possess (Kornblith et al., 2019): invariance with respect to rotations, isotropic scaling while *not* invariant to general invertible linear transforms. In Table 1, we compare CKA, our proposal, and other methods.

2.3.1. Centered Kernel Alignment

A key insight from CKA is that, instead of comparing multivariate features of examples, we can compare the structure of *pairwise similarities* of input instances under the two representations. For simplicity, here we only consider a linear kernel (i.e. dot product) as a measure of similarity. See (Kornblith et al., 2019) for the generalization to any kernels. Given two layer representations X and Y, the pairwise similarities among examples are encoded in the Gram matrices $K = XX^T$ and $L = YY^T$. One can measure the similarity between these two structures by their dot product:

$$\langle K, L\rangle = \sum_{i,j=1}^{n} K_{ij}L_{ij} = tr(KL) = tr(XX^TYY^T)$$

Once X and Y are centralized, denoted by $\hat{X}$ and $\hat{Y}$, the dot product resembles the squared Frobenius norm of cross-covariance matrix between X and Y:

$$\langle \hat{K}, \hat{L}\rangle = tr(\hat{X}\hat{X}^T\hat{Y}\hat{Y}^T) = ||\hat{X}^T\hat{Y}||_F^2$$

CKA further normalize this similarity measure so that it is invariant to isotropic scaling:

$$CKA_{Linear}(X, Y) = \frac{||\hat{X}^T\hat{Y}||_F^2}{||\hat{X}^T\hat{X}||_F||\hat{Y}^T\hat{Y}||_F} \tag{2}$$

In the next session we will show that with a simple change of the norm, we can design another similarity measure which also satisfy all three invariances while having a simpler geometric interpretation.

2.3.2. Orthogonal Procrustes Problems

As we have explained, the Grand Tour smoothly rotates data using a sequence of orthogonal transformations. In other words, the *visualization* is invariant to rotations. Trying to align two rotationally invariant visualizations naturally gives us an orthogonal Procrustes problem (Gower et al., 2004). And the alignment naturally induces a new similarity index, similar to CKA, that is invariant to rotation but not invariant to other invertible linear transformations.

We first introduce the setup of orthogonal Procrustes problems. Following the setup in Section 1, given two representations $X \in \mathbb{R}^{n\times p_1}$ and $Y \in \mathbb{R}^{n\times p_2}$, we further assume $p_1 = p_2 = p$. Orthogonal Procrustes problem looks for an orthogonal transformation $Q \in \mathbb{R}^{p\times p}$ that minimizes the squared euclidean distance between Y and the transformed X:

$$argmin_Q \; ||XQ - Y||_F^2, \quad \text{subject to } Q^TQ = I$$

Table 1: Comparing properties of similarity indices. Our method satisfies all three invariants that Centered Kernel Alignment (CKA) (Kornblith et al., 2019) does. Other methods, such as linear regression, SVCCA (Raghu et al., 2017) and PWCCA (Morcos et al., 2018) fail to satisfy some of these properties. In addition to three invariance guarantees, our method provides a scalable, instance-level and well-aligned visualization of neural network layers.

	Non-invariant to Linear Transform	Invariant to Orthogonal Transform	Invariant to Isotropic Scaling	Scalable	Instance-level Visualization	Alignment
Others	✓/✗/conditional	✓/✗	✓/✗	?	?	?
CKA	✓	✓	✓	✗	✗	?
UMAP + Procrustes (**Ours**)	✓	✓	✓	✓	✓	✓

Geometrically, it means that orthogonal Procrustes problem looks for the best rotation (Q) that aligns X with respect to Y, without scaling X on any dimensions.

To see how orthogonal Procrustes is related to CKA, we follow the note in (Kornblith et al., 2019). The squared Frobenius norm in the minimization can be expanded into

$$||XQ - Y||_F^2 = ||X||_F^2 + ||Y||_F^2 - 2tr(Y^T XQ)$$

The two squared norms are constants, therefore minimizing the squared Frobenius norm is equivalent to maximizing the trace.

$$max_Q\ tr(Y^T XQ), \quad \text{subject to } Q^T Q = I \tag{3}$$

The maximum is given when $Q^* = UV^T$, where U and V are the left and right singular vectors of $X^T Y$, i.e. $U\Sigma V^T = X^T Y$. See (Gower et al., 2004) for the derivation. Plugging the solution $Q^* = UV^T$ into Eq 3 gives

$$max_Q\ tr(Y^T XQ) = tr(\Sigma) = ||X^T Y||_*$$

where $||\cdot||_*$ denotes the nuclear norm. As discussed in the appendix of (Kornblith et al., 2019), this resembles the numerator of linear CKA in Eq 2, with the squared Frobenius norm replaced by a nuclear norm. With this connection in mind, we can define a new similarity measure based on orthogonal Procrustes, s_{op}.

$$s_{op}(X, Y) = \frac{||X^T Y||_*}{\sqrt{||X^T X||_* ||Y^T Y||_*}} \tag{4}$$

This is in the same form of linear CKA in Eq. 2, where every squared Frobenius norm is replaced by a nuclear norm. This similarity index satisfies the three desired invariance properties that CKA possessed. In addition, this similarity index has a straightforward geometric interpretation: it measures the degree of alignment of two configurations after performing the optimal rotation on one of them. More importantly, the alignment associated with this similarity can be used to match *instance-level* visualizations of different layer representations. Table 1 summarizes the advantage of our methods over CKA and other alignment methods discussed in (Kornblith et al., 2019). In UMAP Tour, we use orthogonal Procrustes to align two *dimensionality reduced* representations.

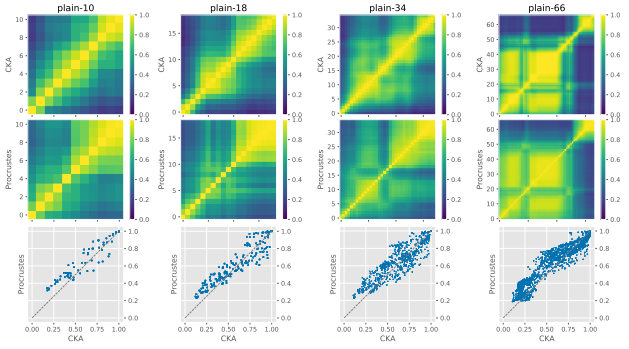

Figure 2: Comparing linear CKA with orthogonal Procrustes. The first two rows show the layer-wise similarity matrix generated by the two mechanisms. We see that orthogonal procrustes, even on the UMAP embedding space, can produce similar scores and show block structures when compared to Linear CKA (between the neuron activation spaces). The bottom row shows their numerical similarity.

2.3.3. Alignment between Embeddings

The UMAP embeddings on two different layers are learned independently and not aligned by default. To better compare two embeddings, we use orthogonal Procrustes to align the two embedded representations. We align the UMAP embeddings, instead of aligning the neuron activation vectors, to scale our method to real-world models. Aligning the embedding space makes the alignment mechanism, in our case orthogonal Procrustes, more practical. Orthogonal Procrustes relies on singular value decomposition of $X^T Y \in \mathbb{R}^{p \times p}$, whose runtime complexity is cubic to the number of features (p) in data (Golub & Van Loan, 2013). UMAP greatly reduces the dimensionality and makes the alignment much faster in practice. As importantly, UMAP is able to capture most of structures in data with manageable loss of preciseness. In Figure 2, we compare the layer-level similarities measured by CKA on the original neuron activations with the ones measured by orthogonal Procrustes on the 15 dimensional UMAP embeddings. We trained 4 all-convolutional nets (Springenberg et al., 2014) with different depths on the CIFAR-10 dataset (Krizhevsky et al., 2009),

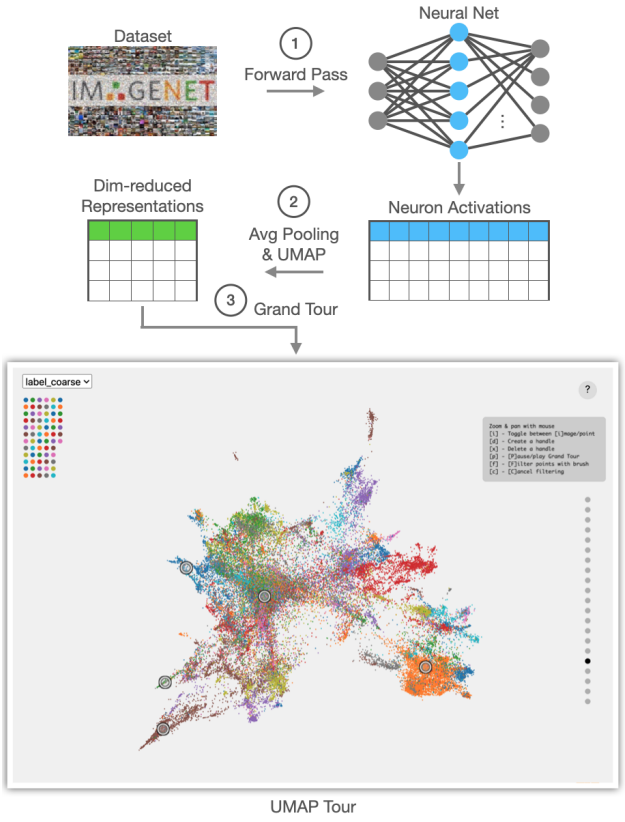

Figure 3: The UMAP Tour pipeline and user interface

configured in a similar way as (Kornblith et al., 2019). Even though UMAP has greatly reduced the dimensionality of data, we see the two mechanisms report similar values and disclose similar block structures in the similarity matrices.

3. Method

The basic steps of UMAP Tour is straightforward: we first take a dataset and pass it to the neural network of interest and record the neuron activations in every layer. Next, for each layer we use UMAP to project the activation to a few (in our experiments, 15) dimensions. Finally, we use the Grand Tour to visualize the embeddings. The three steps are summarized in Figure 3.

3.1. Data Preprocessing

We start by choosing a neural network model of interest and selecting a probing dataset for it. The dataset is meant to serve as a probe to understand the internal behavior of neural network layers. It does not have to be the training or validation set on which the model was trained or validated. For example, even though GoogLeNet (Szegedy et al., 2015) is most commonly trained on ImageNet (Deng et al., 2009), we can either use ImageNet or other image datasets to validate concepts learned by GoogLeNet. This is similar to the setup of testing with concept activation vectors (TCAV) (Kim et al., 2018), but we do not need to explicitly define a concept or be restricted to binary concepts (e.g. 'striped' vs 'non-striped'). In some of the experiment in Section 4, we use a pre-trained GoogLeNet and probe it with the validation split of ImageNet (Deng et al., 2009), while in others we use the entire (training + validation) FairFace (Kärkkäinen & Joo, 2019) dataset.

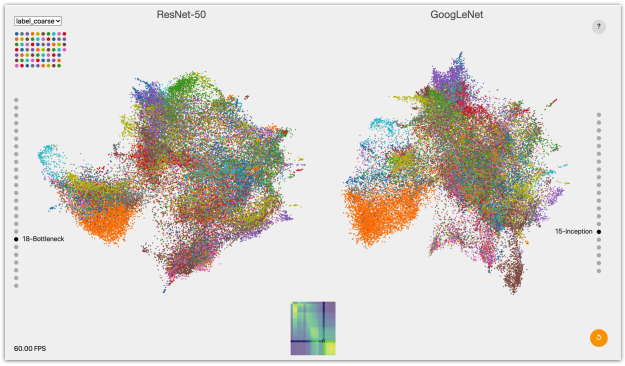

Figure 4: UMAP Tour for comparing two architectures.

As the first step, for each image in the chosen dataset we make a forward pass into the network and store their neuron activations of each layer. By the end of this step, on each layer we have a neuron activation matrix of size $n \times p$, where n is the number of examples in the dataset and each of the p columns corresponds to an output neuron in that layer. The number of neurons in a layer can be very large, usually in the order of hundreds of thousand dimensions for convolutional layers, which makes it hard to visualize in practice. In the third step, we used UMAP to reduce the number of features to a manageable size. We project neuron activations to 15 dimensions in order to capture enough interesting structures in the data. When fitting the UMAP embeddings [1], we keep all hyperparameters as default except for the embedding dimensions ('n_components'). We chose UMAP for its good runtime performance, flexibility and reliability to capture global structures in the data, although method works for other dimensionality reduction techniques. In this specific case, however, UMAP alone is still not practical for handling $\sim$ 100k dimensional neuron activation data. Even though UMAP scales reasonably well with the number of dimensions, it is still time consuming to apply UMAP directly on activations with hundreds of thousand dimensions. To reduce the runtime of UMAP while preserving most of structural features in data, we apply an adaptive average pooling before applying UMAP. In other words, step ② in Figure 3 consists of two steps: we first apply 2D average pooling to reduce the dimension of neuron activations (from hundreds of thousand) to a few tens of

[1] https://umap-learn.readthedocs.io/

thousand, and then apply UMAP on the average-pooled activations. See supplementary for more experimental details, such as the computing infrastructure used, dimension and runtime complexity of each layer.

3.2. Visualization

In the final step of UMAP Tour, we use the Grand Tour (Asimov, 1985) to visualize 'every aspect' of the embedding space using an animated scatter plot. At the bottom of the pipeline (Figure 3) we show our user interface. We used WebGL and D3.js (Bostock et al., 2011) to build our web interface. As an example, we are showing $50k$ images from ImageNet validation set inside the 16-Inception layer of a trained GoogLeNet. The central view shows the Grand Tour view. In this view, user can zoom and pan via mouse sweep. In addition, user can steer the projection via direct manipulation (Li et al., 2020) on data points (see Section 3.2.1). On top left of the interface (Figure 3), user can color data points by attributes via the option menu, such as coloring by label, prediction confidence or prediction error. The color legend will change accordingly. On the right, user can switch to different layers of the model by clicking on the gray dots. The transitions between layers are made simple via orthogonal Procrustes, see Section 3.2.2. To compare two models, we take two Grand Tour views, align them using orthogonal Procrustes and display them side by side (Figure 4). We also draw the layer-wise similarities as a heatmap on the bottom. Clicking on any entry of the similarity matrix quickly switches to the Grand Tour of corresponding *pair* of layers. Other functionalities, such as direct manipulation, are provided via keyboard shortcuts, which are explained on top right of the interface. For example, user can switch to original images by pressing [i] or apply spatial filters by pressing [f].

3.2.1. Direct Manipulation

The Grand Tour, by default, walks through all possible projections in a predetermined, but somewhat random manner. Controlling the projection would let users navigate to the views of their own interest, instead of waiting for them to come through the random walk. Manual control can be done via direct tunning the entries of Grand Tour projection matrix $GT \in \mathbb{R}^{p\times p}$. Specifically, since data points are projected via Eq.1, the first two components in every j^{th} row of the GT determines where each standard basis vector e_j is projected in the plot. Before (Li et al., 2020), the manual control is often done via a separate interface, through which user have to select one of the p variables and change its value by sweeping the mouse. However, controlling each variable not as intuitive as controlling data points. This is especially true for embeddings, since each variable in the UMAP embedding does not usually has semantics. See (Li et al., 2020) for details.

3.2.2. Aligning Representations

Transition between layers: When visualizing one model with UMAP Tour, we switch the view between layers through a smooth and traceable transition. Specifically, when switching from one layer Y to another X, we first align the next layer X against the previous layer Y via orthogonal Procrustes Q^*. Next, we linearly interpolate the two aligned representations: $X(s) = s \cdot XQ^* + (1-s) \cdot Y$. Finally, the interpolation is viewed via Grand Tour projection as usual (Eq.1).

Aligning architectures: When visualizing two models side by side, we align their layer representations in the same manner using orthogonal Procrustes. When comparing the UMAP embedding of one model Y with the embedding X from another model, we show Y and XQ^* side by side in UMAP Tour. When directly manipulation is applied on any one of the views, the change is applied to both views so that the views are kept aligned (Figure 4).

4. Experiments and Results

We first test our method with a single neural network. We used both the dataset for training as well as an external dataset to probe the network. Within a single network, we see GoogLeNet use auxiliary concepts, such as the existence of human in the image classification that are not directly supervised by the labels of the task. With the help of an external face dataset, we learned that GoogLeNet also learns certain attributes of human faces, while not being trained to do so. When comparing two different neural architectures, we find distribution-level similarity as well as *instance-level difference* between their internal representations.

4.1. Layer Dynamics in a Single Model

Since an image classifier is a mapping from raw pixel colors to the label class of the object in the image, one should expect very early layers capture low-level features such as overall color and final layers summarize features into high-level concepts such as image class. The UMAP Tour confirms this layer-to-layer dynamics within the model. Moreover, it reveals how a sequence of intermediate layers approach this goal in multiple steps.

4.1.1. Probing GoogLeNet with ImageNet

When neural network is trained as an image classifier, the neuron activation closer to the raw input should encode low-level features; while in layers close to the final prediction, the neuron activations should capture the image class. This end-to-end behavior is a direct consequence of neural networks being universal approximators, and one should expect this without the help of UMAP Tour. However, it is typically hard to observe how the concepts are gradually

Figure 5: Concepts learned in 12-Inception layer of GoogLeNet: ①② Orientation of dogs; ③ animal faces; and ④ human faces.

learned in the intermediate layers. UMAP Tour sheds some light onto this. Through UMAP Tour, we found that in the 12-Inception layer of GoogLeNet encode the orientation of animals (mainly dogs) despite being only trained to classify them (Figure 5). In the same layer, the network also recognizes human faces, despite no human label is provided during training. When classifying objects that may or may not appear with human (e.g. dogs alone or with their owners, fishes in the wild or held by humans), GoogLeNet considers the two cases separately (Figure 7). Diving down from 12-Inception layer to a deeper global pooling layer (17-AdaptiveAvgPool2D), we observed a small group of dog-with-human images gradually moves toward the main cluster of dogs. On the other hand, the cluster of images showing fish caught by human remains stable in this process. They only move to the cluster of fishes in the penultimate fully connected layer (19-Linear). See supplementary video for the animated demonstration of this behavior. This behavior confirms our believe that UMAP Tour helps identify patterns in each layer, which can be particularly useful in, for instance, finding the best layer for fine-tuning or transfer learning.

4.1.2. Probing GoogLeNet with FairFace

In the previous section we found a human detector in GoogLeNet. This new finding lead to more thorough analysis. In this case, we are interested in how this human detector works in terms of what patterns it captures and whether it biases toward certain race, gender, or age groups. Since ImageNet do not have these attributes associated with the images, we use an external human face dataset, FairFace (Kärkkäinen & Joo, 2019), for a deeper inspection.

FairFace contains human faces of various orientations, surroundings and lighting conditions. For example some people wear sunglasses or stand behind microphones, and we can see through UMAP Tour that GoogLeNet is able to

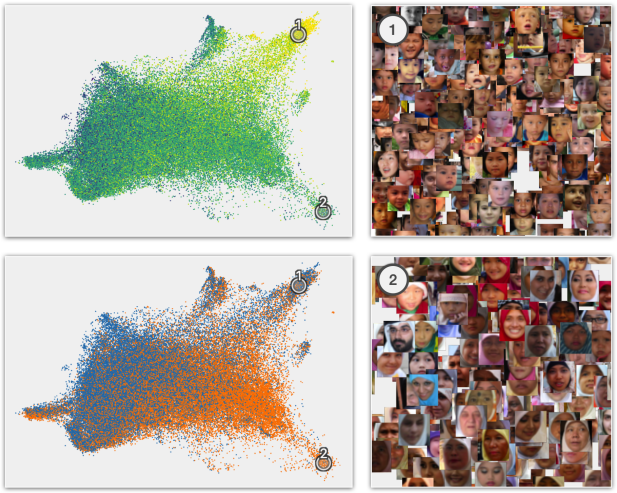

Figure 6: Coloring data points by attributes from FairFace dataset reveals GoogLeNet's ability to identify certain age and gender groups. In the 16-Inception layer, **top:** coloring by age (from ● age 0-2 to ● more than 70) shows 0-2 age group in one cluster; **bottom:** coloring by gender (● males ● females) exhibits veiled women in another cluster.

recognize the apparel or device in such cases. In addition, examples in FairFace are labeled with three attributes: race, gender and age groups. In UMAP Tour, we color data points by these attributes to see if any activation pattern is related to these attributes. We found that GoogLeNet uses face to help recognize certain apparel or devices, and sometimes apparel can be related to certain age or race groups. For example, in 16-Inception layer, GoogLeNet identifies baby faces and woman in veils, as shown in Figure 6. This is not very surprising given the distinctive look of two groups. For most pictures of veiled women, GoogLeNet predicted the surrounded clothings as neck brace, abaya or bonnet, etc. Similarly, baby faces have been predicted to similar clothings around their faces, such as bonnet, bib or bath towel.

We did not find any global-scale pattern that is related to certain racial or ethnic groups. This is not very surprising because GoogLeNet was not trained on face related tasks and FairFace well-balanced on every ethnic groups. However, since certain apparel such as veil is closely related to religious customs and a face dataset typically does *not* sample uniformly across race/ethnic groups *conditioned* on these apparel, a globally well-balanced dataset might not distribute evenly in some local neighborhood in GoogLeNet internal layers. For example, a large proportion of veils are worn by women from Southeast Asian or Middle Eastern, while greater proportion of recognized sunglasses are worn by white people. These observations may certify concerns around some reckless claims of pre-trained network

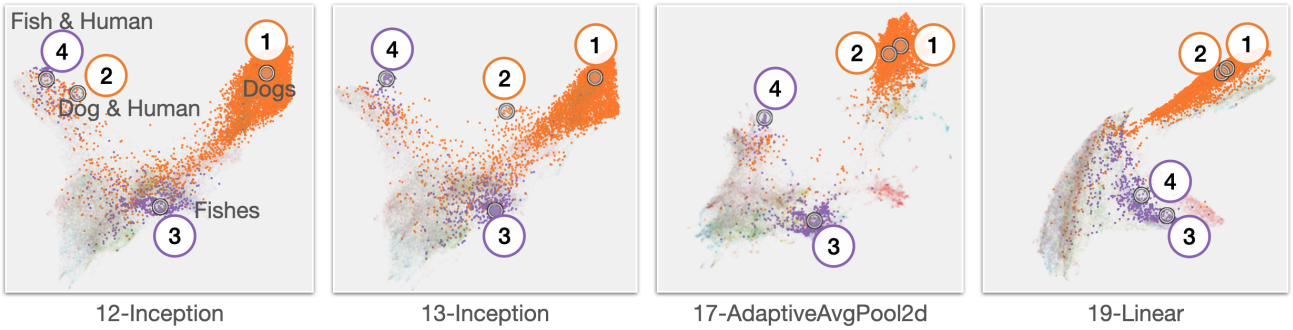

Figure 7: GoogLeNet consider humans during classification of other objects. From the 12-Inception layer to 17-AdaptiveAvgPool2d, a group of dog-with-human images (②) moves from the main cluster of humans to the main cluster of ● dogs (①), while the fish-with-human images (④) remain stable in the view. The fish-with-human images (④) move closer to the cluster of other ● fishes (③) only after the penultimate (19-Linear) layer.

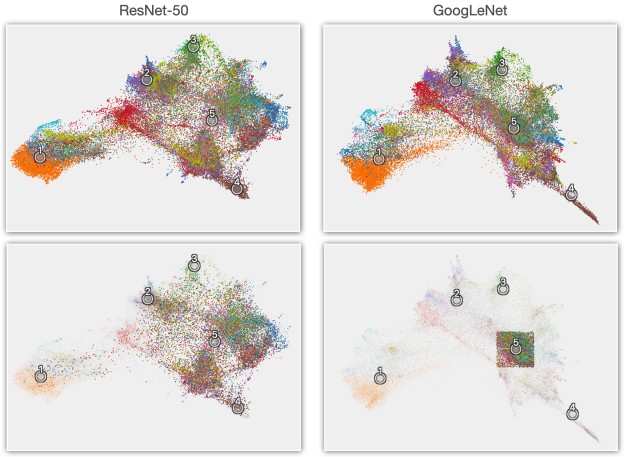

Figure 8: Comparing ResNet-50 (18-Bottleneck layer) with GoogLeNet (15-Inception layer). The two layers have a similarity score of 0.810.**Top:** The two views match in multiple landmarks, such as ① dogs, ② bugs and snakes, ③ food and ④ vehicles. **Bottom:** However, filtering by a box around ⑤ reveals GoogLeNet's unique cluster of (electronics, buildings, clothings, etc) which is more distributed in ResNet-50.

in transfer learning. For example, when someone claims to be able to recognize religious belief of people by fine tuning GoogLeNet, the network might simply take spurious features, such as veils, as signals. In such cases, UMAP Tour is able to point out the problem with a direct illustration of the neural network internals. See supplementary material for more figures of GoogLeNet capturing orientation of faces, different lighting conditions, etc.

4.2. Comparing Two Models

Although many works has been done to visually interpret the internal of neural networks (Olah et al., 2017; Nguyen et al., 2019; Selvaraju et al., 2017), fewer of them is able to directly compare two architectures. UMAP Tour is designed for comparison especially when equipped with orthogonal Procrustes. For example, ResNet-50 and GoogLeNet have a similar index 0.810 on a pair of intermediate layers. In UMAP Tour, we see similar and well-aligned UMAP embeddings in these layers, as shown by some landmarks (①-④ in Figure 8). In addition to the similarity, UMAP Tour also shows the *dissimilarity* between the two: GoogLeNet as a unique cluster of man-made objects (electronics, buildings, clothings, etc) which is more distributed in ResNet-50 (⑤ in Figure 8). In other words, UMAP Tour not only uses the similarity as a measure to align two representations, but also explains their *dissimilarities* in its instance-level visualizations.

5. Conclusion

In this work we explored how classical visual method (the Grand Tour) can be combined with modern embeddings (namely UMAP) to reason about a single deep neural network or compare two neural architectures. We build UMAP Tour and use it to visualize patterns within state-of-the-art models such as GoogLeNet and ResNet-50. The alignment method used in this visualization, namely orthogonal Procrustes, naturally induces a new similarity measure between neural network layers. Theoretically, the orthogonal Procrustes similarity itself is a linear measure but we use it on the UMAP embeddings - a product of approximating a non-linear kernel. Therefore one may be able to frame the whole process as a CKA-like similarity with a special kernel. Empirically, this similarity measure has comparable behavior with CKA, while the alignment gives us an intuitive, instance-level visualization. Our visualization built around these notions has revealed several less well-known patterns in the neural network internals and shed some light on the transparency and fairness concerns of neural networks.

References

Ancona, M., Oztireli, C., and Gross, M. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International Conference on Machine Learning*, pp. 272–281. PMLR, 2019.

Asimov, D. The grand tour: a tool for viewing multidimensional data. *SIAM journal on scientific and statistical computing*, 6(1):128–143, 1985.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Borg, I. and Groenen, P. J. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

Bostock, M., Ogievetsky, V., and Heer, J. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):23012309, December 2011. ISSN 1077-2626. doi: 10.1109/TVCG.2011.185. URL https://doi.org/10.1109/TVCG.2011.185.

Carter, S., Armstrong, Z., Schubert, L., Johnson, I., and Olah, C. Exploring neural networks with activation atlases. *Distill.*, 2019.

Cook, D. and Buja, A. Manual controls for high-dimensional data projections. *Journal of computational and Graphical Statistics*, 6(4):464–480, 1997.

Cook, D., Buja, A., Cabrera, J., and Hurley, C. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3):155–172, 1995.

Cook, D., Buja, A., Lee, E.-K., and Wickham, H. Grand tours, projection pursuit guided tours, and manual controls. In *Handbook of data visualization*, pp. 295–314. Springer, 2008.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL http://www.image-net.org/.

Goldfeld, Z., Berg, E. v. d., Greenewald, K., Melnyk, I., Nguyen, N., Kingsbury, B., and Polyanskiy, Y. Estimating information flow in deep neural networks. *arXiv preprint arXiv:1810.05728*, 2018.

Golub, G. and Van Loan, C. Matrix computations 4th edition the johns hopkins university press. *Baltimore, MD*, 2013.

Gower, J. C., Dijksterhuis, G. B., et al. *Procrustes problems*, volume 30. Oxford University Press on Demand, 2004.

Kärkkäinen, K. and Joo, J. Fairface: Face attribute dataset for balanced race, gender, and age. *arXiv preprint arXiv:1908.04913*, 2019. URL https://github.com/dchen236/FairFace.

Karpathy, A. t-sne visualization of cnn codes, 2012. URL https://cs.stanford.edu/people/karpathy/cnnembed/.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C. J., Wexler, J., Viégas, F. B., and Sayres, R. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2673–2682. PMLR, 2018.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. E. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Li, M., Zhao, Z., and Scheidegger, C. Visualizing neural networks with the grand tour. *Distill*, 5(3):e25, 2020.

McInnes, L. Performance comparison of dimension reduction implementations, 2018. URL https://umap-learn.readthedocs.io/en/latest/benchmarking.html.

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

Morcos, A. S., Raghu, M., and Bengio, S. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 5732–5741, 2018.

Nguyen, A., Yosinski, J., and Clune, J. Understanding neural networks via feature visualization: A survey. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 55–76. Springer, 2019.

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

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

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6076–6085, 2017.

Rauber, P. E., Fadel, S. G., Falcao, A. X., and Telea, A. C. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):101–110, 2016.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500): 2323–2326, 2000.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 618–626. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.74.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Sun, Y., Chen, Y., Wang, X., and Tang, X. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 1988–1996, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 1–9. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298594.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Wold, S., Esbensen, K., and Geladi, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.