

Studying CNN representations through activation dimensionality reduction and visualization

by

Nolan Simran Dey

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master's of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2021

© Nolan Simran Dey 2021

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis by articles contains two articles which I co-authored.

The first article is contained in Chapter 4.

Article details: Identifying and interpreting tuning dimensions in deep networks. In *Shared Visual Representations in Human & Machine Intelligence NeurIPS Workshop, 2020*. Nolan S. Dey, J. Eric Taylor, Bryan P. Tripp, Alexander Wong, Graham W. Taylor

Personal contributions: The idea to study the tuning dimensions of DNNs came jointly from Bryan P. Tripp, Alexander Wong, and Graham W. Taylor. As the first author, I was responsible for the programming, experimentation, and writing of the majority of the manuscript, under the supervision of my four co-authors.

The second article is contained in Chapter 6.

Article details: The surprising effectiveness of PCA for understanding CNN representations. *Submitted to NeurIPS 2021 for review*. Nolan S. Dey, J. Eric Taylor, Bryan P. Tripp, Alexander Wong, Graham W. Taylor

Personal contributions: As the first author, I was responsible for the programming, computational experimentation, and the writing of the majority of the manuscript, under the supervision of my four co-authors. In addition to providing supervision, the second author J. Eric Taylor was responsible for the user study. He conceived the experimental design, applied for the IRB, recruited participants, conducted the study, and analyzed the results.

Abstract

The field of explainable artificial intelligence (XAI) aims to explain the decisions of DNNs. Complete DNN explanations accurately reflect the inner workings of the DNN while interpretable explanations are easy for humans to understand. Developing methods for explaining the representations learned by DNNs that are both complete and interpretable is a grand challenge in the field of XAI.

This thesis makes contributions to the field of XAI by proposing and evaluating novel methods for studying DNN representations. During forward propagation, each DNN layer non-linearly transforms the input space in some way that is useful for minimizing a loss function. To understand how DNNs represent their inputs, this work develops methods to examine each DNN layer’s activation space.

The first article contributes an unsupervised framework for identifying and interpreting “tuning dimensions” in the activation space of DNN layers. The method consists of fitting a dimensionality reduction model to a layer’s activations, then visualizing points along the axes defined by each of the reduced dimensions. The method correctly identifies the tuning dimensions of a synthetic Gabor filter bank, and those of the first two layers of InceptionV1 trained on ImageNet.

The second article builds upon the first article with a simple and greatly improved visualization method that enables studying every layer of AlexNet. Through a quantitative comparison, the article demonstrates that the principal component analysis (PCA) basis for activation space offers more complete and more interpretable explanations than the traditional neuron basis. This thesis provides deep learning researchers with tools to better understand the representations learned by DNNs.

Acknowledgements

I am incredibly fortunate to have an amazing support network that made writing this thesis possible.

First I would like to thank my parents Amrit Malhotra and Tito Dey, who made big sacrifices to give me a privileged upbringing and allow me to pursue my dreams.

Thank you to Bryan Tripp, Graham Taylor, Alexander Wong, and Eric Taylor for being excellent mentors. You pushed me to be a better researcher, offered advice to keep me on the right path, and made me feel like I belong in the daunting world of research. I will always be grateful for your mentorship and encouragement.

To Xueyang Yao, Salman Khan, Ramashish Gaurav, Parsa Torabian, Rajan Iyengar, and Victor Osorio from the BRAIN lab at the University of Waterloo, and Magdalena Sobol, Terrance DeVries, Vikram Voleti, Yani Ioannou, Kenyon Tsai, Brendan Duke, Shashank Shekhar, Angus Galloway, Michal Lisicki, Mahmoud Gamal, Sara El-Shawa, Kristina Kupferschmidt, Boris Knyazev, Eu Wern Teh, Rylee Thompson, Sam Motamed, Carolyn Augusta, Elahe Ghalebi, Ethan Jackson, and Stefan Schneider from the MLRG at the University of Guelph, thank you for being great friends, colleagues, and mentors. Magdalena Sobol deserves an extra shout out because she provided invaluable writing feedback for both the articles in this thesis.

I am also very grateful to BMO Bank of Montreal for funding my thesis and encouraging progress in explainable artificial intelligence research.

Thank you to the Vector Institute for providing world-class computational resources. They allowed me to iterate much more quickly and greatly reduced my stress.

Finally, I would like to thank Hai-Dao Le-Nguyen for her unfaltering love and support.

Table of Contents

List of Figures	x
List of Tables	xiv
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Contributions	2
2 Background	4
2.1 Deep neural networks	4
2.1.1 Fully-connected layers	4
2.1.2 Convolutional neural networks	5
2.1.3 Parameter estimation	6
2.1.4 Biological motivations	6
2.2 Explainable artificial intelligence	7
2.2.1 Taxonomy of explainability methods for deep neural networks	7
2.2.2 Visualizing DNN activations in input space	9
2.2.3 Supervised explainability methods	10
2.2.4 Unsupervised explainability methods	11
2.2.5 Manual studies of DNN representations	12

2.2.6	Understanding the activation space of generative models	13
2.2.7	Human XAI studies	13
2.2.8	Connection to neuroscience	13
3	Prologue to first article	16
3.1	Article details	16
3.1.1	Personal contributions	16
3.2	Context	16
3.3	Contributions	17
4	Identifying and interpreting tuning dimensions in deep networks	18
4.1	Introduction	18
4.2	Related work	19
4.3	Method	20
4.3.1	Collecting activations	20
4.3.2	Identifying tuning dimensions	20
4.3.3	Visualizing points along each tuning dimension	21
4.3.4	Interpreting tuning dimension visualizations	21
4.4	Results	22
4.4.1	Identifying tuning dimensions of a synthetic Gabor bank	23
4.4.2	Identifying tuning dimensions of InceptionV1 layers	24
4.5	Conclusions and future work	25
5	Prologue to second article	27
5.1	Article details	27
5.1.1	Personal contributions	27
5.2	Context	27
5.3	Contributions	29

6	The surprising effectiveness of PCA for understanding CNN representations	30
6.1	Introduction	30
6.2	Method	31
6.2.1	Sample a layer’s activations	32
6.2.2	Identify principal components	33
6.2.3	Visualize points along principal components	33
6.2.4	Interpret visualizations	36
6.3	Results	36
6.3.1	Examples of principal component visualizations	36
6.3.2	Measures of completeness	37
6.3.3	Explanation Interpretability: Human Validation User Study	39
6.4	Limitations	42
6.5	Conclusion	43
7	Conclusions	44
7.1	Future work	44
	References	47
	APPENDICES	56
A	First article appendix	57
A.1	Comparing sampling methods	57
A.2	Gabor bank construction	57
A.3	Gabor bank tuning dimension visualizations	58
A.4	InceptionV1 tuning dimension visualizations	58

B	Second article appendix	68
B.1	AlexNet architecture	68
B.2	Masking natural image regions that contribute strongly to activation similarity	68
B.3	Comparison of methods for sampling points along basis vectors	70
B.4	Computational resources	70
B.5	Distribution of principal component weights	71
B.6	Additional User Study Details	71
B.7	Neuron ablation	72
B.8	Neuron visualizations	72
B.9	Additional principal component visualizations	72

List of Figures

3.1	Toy example of tuning curves for a population of neurons. The tuning dimension of this population is the stimulus orientation.	17
4.1	Tuning dimension visualizations of the first layer of InceptionV1 (conv2d0). Top: Visualized points along first ICA component show increasing frequency of vertically oriented edges. Bottom: Visualized points along sixth ICA component show dark/light edges switching to light/dark edges which is associated with a change in Gabor filter phase. 32 points were visualized for each tuning dimension but to improve readability, the most informative 6 are shown. Best viewed in color.	22
5.1	Task presented to participants of the user study.	28
6.1	Overview of our method. We sample \mathbb{R}^D activations from a layer’s activation map (1), then identify the principal components of the layer’s activation space (2). Finally we visualize points along each principal component (3) and interpret the visualizations (4).	32
6.2	Examples of synthetic receptive field (Synthetic-RF) [54], synthetic full size (Synthetic-Full) [54], and natural receptive field (Natural-RF). Although Synthetic-Full has a higher resolution than Synthetic-RF and Natural-RF, each of the visualizations is scaled to be the same size.	34
6.3	The full visualization for the fourth principal component of conv4. Positive values along this component correspond with the presence of a fur texture, while negative values seem to correspond with well-defined straight edges in the foreground and foliage in the background. In the far right columns, each of the 3 visualization types provides a different visualization of fur.	37

6.4	Natural-RF visualizations along the first 6 principal components of conv1. Brightness (PC 1), phase (PC 2,3), color contrast (PC 4,5), color-center-surround (PC 6).	38
6.5	Cumulative sum of explained variance for each AlexNet convolutional layer. Both PCs and neurons are ordered by descending variance.	39
6.6	For each of AlexNet’s five convolutional layers, we ablate basis vectors in activation space and measure the effect on ImageNet top-1 validation accuracy. "PC order" (blue line) and "PC reverse order" (orange line) ablated PCs in descending and ascending order of eigenvalues respectively. "Neuron order" (green line) and "Neuron reverse order" (red line) ablated neurons in descending and ascending order of their individual effect on validation accuracy respectively. The points where PCs are responsible for specific percentages of explained variance are also annotated.	40
6.7	User study results for PCA (top) and neuron (bottom) basis visualizations.	42
A.1	Visualizations of the first ICA component fit to activations from the first layer of InceptionV1 (conv2d0). Top: Points sampled uniformly between the observed minimum and maximum values along the component. Bottom: Points sampled such that the same proportion of observed points in activation space lie between each of the points along the component. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	58
A.2	Top: Gabor bank filter weights. The values in each of the RGB channels are equal. Bottom: Preferred stimuli of each Gabor filter found through feature visualization [54]. Best viewed in color.	60
A.3	Gabor bank PCA-3 and PCA-6 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Due to the ordering of PCA, the first 3 principal components are shared between PCA-3 and PCA-6. Best viewed in color.	61
A.4	Gabor bank ICA-3 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	61

A.5	Gabor bank ICA-6 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	62
A.6	Gabor bank NMF-3 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. No meaningful variation was observed along each tuning dimension so no tuning dimensions were interpreted. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	62
A.7	Gabor bank LLE-3 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. No apparent pattern was observed along each tuning dimension so no tuning dimensions were interpreted. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	63
A.8	conv2d0 PCA-16 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	64
A.9	conv2d0 ICA-16 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	65
A.10	conv2d1 PCA-16 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	66
A.11	conv2d1 ICA-16 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.	67

B.1	Top: Histogram of activations from the ImageNet training set, projected onto the second PC (PC 2) of conv1. The interval defined by the minimum and maximum is shown in green. The interval defined by the first and ninety-ninth percentile is shown in red. Middle and bottom: Visualizations of points along PC 2 of conv1, uniformly sampled between the first and ninety-ninth percentile (middle), and uniformly sampled between the minimum and maximum (bottom).	73
B.2	Histograms of principal component weights for each AlexNet layer. The kurtosis for each distribution is also included in each plot title. A kurtosis of 0 corresponds to a normal distribution. From conv1 to conv5, the distribution of PC weights becomes progressively more similar to a normal distribution. Then, fc1 has a high kurtosis but from fc1 to fc3, the distribution of PC weights once again becomes progressively more similar to a normal distribution.	74
B.3	Full text instructions presented to participants of the user study.	75
B.4	For each of AlexNet’s five convolutional layers, we ablate neurons individually and measure the effect on ImageNet top-1 validation accuracy. Neurons were individually ablated by setting their activations to zero during the forward pass.	76
B.5	Natural-RF visualizations along the first 6 neurons of conv1.	76
B.6	Natural-RF visualizations along the first 6 neurons of conv2.	77
B.7	Natural-RF visualizations along the first 6 neurons of conv3.	77
B.8	Natural-RF visualizations along the first 6 neurons of conv4.	78
B.9	Natural-RF visualizations along the first 6 neurons of conv5.	78
B.10	Natural-RF visualizations along the first 6 principal components of conv2.	79
B.11	Natural-RF visualizations along the first 6 principal components of conv3.	79
B.12	Natural-RF visualizations along the first 6 principal components of conv4.	80
B.13	Natural-RF visualizations along the first 6 principal components of conv5.	80
B.14	Natural-RF visualizations along the first 6 principal components of fc1. . .	81
B.15	Natural-RF visualizations along the first 6 principal components of fc2. . .	81
B.16	Natural-RF visualizations along the first 6 principal components of fc3. . .	82

List of Tables

4.1	Comparison of identified Gabor bank tuning dimensions. PCA-3 denotes PCA with 3 components.	23
4.2	Comparison of PCA and ICA for identifying tuning dimensions in the InceptionV1 layers <code>conv2d0</code> and <code>conv2d1</code> . PCA-16 denotes PCA with 16 components.	25
B.1	Architecture for the PyTorch [57] implementation of AlexNet [39]. In the parameters column, <code>k</code> , <code>s</code> , <code>p</code> , and <code>d</code> are kernel size, stride, padding, and dropout probability respectively.	69

Chapter 1

Introduction

Deep neural networks (DNNs) have revolutionized computer vision [39, 26], natural language processing [15], and reinforcement learning [69] by achieving state of the art performance in many challenging tasks. Despite the predictive power of DNNs, their adoption in sensitive domains such as transportation, medicine, and finance has been slow because these domains require models whose outputs can be understood and trusted by humans.

1.1 Motivation

One major drawback of DNNs is their “black box” nature; it is hard to understand how these networks arrive at their outputs. The field of explainable artificial intelligence (XAI) develops methods to explain how DNNs arrive at their outputs [23]. A major obstacle to safe adoption of deep neural networks (DNNs) in sensitive applications is the lack of global DNN explanation methods that are both complete (accurately explain a DNN’s function) and interpretable (humans can understand the explanations) [23, 24, 41, 16]. Developing an explanation method for DNNs that is both complete and interpretable is difficult because a trade off exists between completeness and interpretability [23]. A highly complete explanation requires a vast amount of complex information to be presented to a user, whereas a highly interpretable explanation requires a simple explanation with little information.

Popular DNN explanation methods often make choices that increase interpretability at the expense of completeness. For example, by studying individual neurons [47, 50, 51, 54, 55, 5, 49, 6], the distributed nature of deep representations [55, 19, 61, 20, 4, 41, 32] is

disregarded. Likewise, looking for overlap between a network’s representation and a set of user-defined concepts [5, 19, 49, 36, 61] negatively affects explanation completeness; a user is unlikely to define the full set of concepts represented by a DNN.

The field of neuroscience has a similar goal to XAI: developing methods for understanding biological neural networks (BNNs) in the brain. Neuroscientists have developed various approaches for understanding BNN function that may have applications to XAI. One example is looking for a low-dimensional manifold in activation space that accounts for much of the variance [30, 13, 20, 65, 72]. Paths along this low-dimensional manifolds in activation correspond to changes in particular attributes of input stimuli. These methods are well established in neuroscience and have produced many insights into brain function. They are likely to work even better in deep networks, because while it is time-consuming and expensive to record activity in biological neurons, recording millions of activations is computationally cheap.

1.2 Objectives

In this thesis, the primary research question is:

How can meaningful low-dimensional manifolds in the activation space of DNN layers be automatically identified and used to produce both complete and interpretable explanations of DNN representations?

This thesis aims to provide a simple and effective method of identifying low-dimensional manifolds in activation space that capture much of the activation variance. Furthermore, this thesis explores how low-dimensional manifolds in activation space can be used to provide complete and interpretable explanations of DNNs.

1.3 Contributions

The contributions of this thesis are as follows:

- Propose a simple and effective unsupervised framework for applying standard dimensionality reduction methods to DNN activations to identify a DNN
- Proposes a method for interpreting how the variance along a path in activation space relates to variance in input space, leveraging several existing visualization techniques

- Evaluate the effectiveness of principal component analysis (PCA), independent component analysis (ICA) [31], non-negative matrix factorization (NMF) [12], and locally linear embedding (LLE) [62] for identifying meaningful low-dimensional manifolds in the activation space of a synthetic Gabor filter bank, and in the activation spaces of the first two layers of InceptionV1 [73] trained on ImageNet [14]. PCA and ICA were the most effective dimensionality reduction methods for automatically identifying tuning dimensions.
- Demonstrate that PCA can find a significantly more compact and meaningful set of basis vectors when fit to activations before a layer’s non-linearity is applied rather than afterwards
- Extensively study the PCA basis and neuron basis for all layers of AlexNet [39] pretrained on ImageNet [14]. It is shown that the principal components in activation space capture semantically meaningful concepts at each layer. Furthermore, the completeness of the PCA basis and the neuron basis are evaluated through the fraction of explained activation variance and the fraction of validation accuracy lost by ablating basis vectors. It is found that the PCA basis is significantly more complete than the neuron basis. Furthermore, through a user study it is measured that explanations of the top principal components are more interpretable than explanations of the most important neurons. Thus, this thesis demonstrates that the PCA basis offers both a more complete and a more interpretable explanation of AlexNet’s activation space than the neuron basis

Chapter 2

Background

This thesis explores methods for studying the representations of convolutional neural networks applied to supervised image classification tasks. This chapter briefly explains deep neural networks and convolutional neural networks. Then several explainable artificial intelligence techniques for studying the representations of convolutional neural networks are reviewed. This chapter provides additional background context for the articles in this thesis.

2.1 Deep neural networks

Deep neural networks (DNNs) are a type of machine learning model that consist of multiple neural network layers [44]. This section describes two layer types: fully-connected layers and convolutional layers.

2.1.1 Fully-connected layers

The fully-connected layer is the most simple type of DNN layer. A DNN consisting entirely of fully-connected layers can also be called a multi-layer perceptron or fully connected neural network [29]. Each neuron in the current layer is connected to every neuron in the previous layer via a scalar synaptic weight. More concretely, let n be batch size or the number of data points being processed at once, and d^l be the number of neurons in the l^{th} layer. Equation 2.1 defines a fully-connected DNN layer where $a^l \in \mathbb{R}^{n \times d^l}$ is the output or activation of the l^{th} layer, $W^l \in \mathbb{R}^{d^{l-1} \times d^l}$ is the weight matrix, $a^{l-1} \in \mathbb{R}^{n \times d^{l-1}}$ is the output

of the previous layer, $b^l \in \mathbb{R}^d$ is a bias term, and σ is a differentiable nonlinear activation function. The addition of a non-linear activation function enables DNNs to learn highly complex non-linear functions. The most common activation function is the rectified linear unit (ReLU) defined in equation 2.2.

$$a^l = \sigma(a^{l-1}W^l + b^l) \quad (2.1)$$

$$\sigma_{ReLU}(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \quad (2.2)$$

2.1.2 Convolutional neural networks

Convolutional neural networks (CNNs) [43] are a type of neural network containing convolutional layers that possess an inductive bias suited to data where there is a relationship between neighbouring elements in a data instance. For instance, points in a time series are temporally related to one another. Likewise, neighbouring pixels in an image are spatially related (e.g. pixels that define an edge have high contrast with other nearby pixels). A two-dimensional CNN layer is defined in Equation 2.3. Let d^l , h^l , and w^l be the number of channels, height, and width respectively of the l^{th} layer's activations. Let k_h^l and k_w^l be the height and width respectively of the l^{th} layer's convolutional kernel. Let $a^l \in \mathbb{R}^{n \times d^l \times h^l \times w^l}$ be the output or activation of the l^{th} layer, $W^l \in \mathbb{R}^{d^l \times d^{l-1} \times k_h^l \times k_w^l}$ be the weight matrix that defines the l^{th} layer's d^l convolutional filters, $a^{l-1} \in \mathbb{R}^{n \times d^{l-1} \times h^{l-1} \times w^{l-1}}$ be the output of the previous layer, $b^l \in \mathbb{R}^{d^l}$ be the bias term, and σ be a differentiable nonlinear activation function such as ReLU.

$$a_{i,c,j,k}^l = \sigma \left(\sum_{q=1}^{d^{l-1}} \sum_{r=-\lfloor k_h^l/2 \rfloor}^{\lfloor k_h^l/2 \rfloor} \sum_{s=-\lfloor k_w^l/2 \rfloor}^{\lfloor k_w^l/2 \rfloor} a_{i,q,j+r,k+s}^{l-1} W_{c,q,r,s}^l + b_c^l \right) \quad (2.3)$$

When activations from the previous layer a^{l-1} are convolved with the kernels W^l , each element of the output a^l is only a function of the pixels within a $k_h^l \times k_w^l$ neighbourhood. In contrast, for fully-connected neural network layers, each element of the layer output a^l is a function of every element for that data instance. In this way, convolutional layers are locally-connected rather than fully-connected. The locally-connected nature of CNNs reduces the computational cost of processing high dimensional inputs compared to fully-connected DNNs. The same convolutional kernel weights are applied to every spatial position in a^{l-1} ,

meaning that the parameters W^l are shared between every spatial position. This parameter sharing greatly reduces the memory cost of storing parameters and reduces overfitting [43].

2.1.3 Parameter estimation

When a DNN is first created, its parameters W^l and b^l are randomly initialized with a scheme such as ($W^l \sim \mathcal{N}(\mu = 0, \sigma = 1)$ and $b^l = \vec{0}$). A DNN's parameters must then be adjusted from their initialization point for the DNN to estimate a particular function. For a L layer neural network, the network's output $\hat{Y} = a^L$ and the network's input $X = a^0$. A neural network can be written as a function $\hat{Y} = f(W, b, X)$. When applying a DNN to a supervised learning problem, the model must predict the true output Y given an input X . The model's performance is quantified by a differentiable loss function $\mathcal{L}(Y, \hat{Y})$. The backpropagation algorithm [63] exploits the fact that the neural network function f is differentiable to calculate the gradient of the loss function \mathcal{L} with respect to each layer's parameters W^l and b^l : $\frac{\partial \mathcal{L}}{\partial W^l}$ and $\frac{\partial \mathcal{L}}{\partial b^l}$ respectively. These gradients enable the loss function \mathcal{L} to be iteratively minimized using the gradient descent algorithm. The parameter update rules for each gradient descent iteration are shown in Equations 2.4, 2.5, where α is the scalar learning rate.

$$W^l \leftarrow W^l - \alpha \frac{\partial \mathcal{L}}{\partial W^l} \tag{2.4}$$

$$b^l \leftarrow b^l - \alpha \frac{\partial \mathcal{L}}{\partial b^l} \tag{2.5}$$

During forward propagation, each DNN layer non-linearly transforms the input space in some way that is useful for minimizing a loss function. Each layer's activations a^l form a **deep representation** of the input data X . Suppose X is an image and the l^{th} layer of a DNN is a 500 neuron fully connected layer. By forward passing the image X through the DNN up to the l^{th} layer, the activations a^l are a 500-dimensional vector representation of the image X .

2.1.4 Biological motivations

The DNNs discussed thus far are also known as artificial neural networks (ANNs). ANNs bear some similarity to the biological neural networks (BNNs) in the brain. Both ANNs and BNNs are composed of simple units of computation called neurons that are connected

via synaptic weights that can be adjusted to learn functions. However, ANNs are not exact models of BNNs. One key difference is that biological neurons output activation spikes rather than continuous values like artificial neurons. In ANNs, the temporal information in neuron spiking patterns is discarded by summarizing a neuron’s output as a firing rate that can take any non-negative continuous value. Despite the differences between ANNs and BNNs, studies have shown that CNNs can approximate the responses of visual cortices in the brain [75, 84, 9].

2.2 Explainable artificial intelligence

Safety-critical applications of artificial intelligence require transparent models that experts can understand and trust. Explainable artificial intelligence (XAI) is a research area concerned with developing methods to explain artificial intelligence algorithms.

2.2.1 Taxonomy of explainability methods for deep neural networks

The problem of explaining a DNN’s function can be tackled from many different angles. This section provides a taxonomy to illustrate the unique advantages, disadvantages, and trade-offs associated with existing approaches to explainability.

Evaluating XAI methods with completeness and interpretability

XAI methods can be evaluated with two abstract metrics: completeness and interpretability [23]. Completeness describes how accurately an explanation describes a model’s function. Completeness is an abstract quantity that is usually defined specifically for each method. For example, completeness can be measured by the fraction of a DNN that can be explained by a method [5], the alignment of explanations with concepts [19, 60, 70], or the amount of variance in a DNN’s activations that is explained (see Section 6.3.2). Interpretability describes how understandable an explanation is to humans. Since interpretability measures how well humans can understand an explanation, it is typically measured via a user study. In Section 2.2.7 several XAI user studies that measure interpretability are reviewed. DNNs are incredibly complex models. A highly complete explanation will present a great deal of information to properly communicate the complexity of a DNN’s decisions. In contrast, a highly interpretable explanation will present a relatively small amount of information that is

easy to understand. With explainability methods, there is a tradeoff between interpretability and completeness based on the amount of information presented to a user [23].

Instance-specific versus global methods

Instance-specific methods [70, 60, 46, 79, 40] help explain why a model produced a particular output for some specific input data instance. Instance-specific methods are well suited to explaining a model's decision in specific scenarios such as explaining why a model made a particular medical diagnosis. On the other hand, global methods help explain the overall function of the model and are not specific to any particular input. Instance-specific methods offer a less complete explanation of a network than global methods because explanations for particular instances may not generalize well to very different instances [70, 60, 46, 79, 40]. One could study many separate instance-specific explanations but still not gain a global understanding of a DNN's function that generalizes to any data instance. Studying networks with global explanation techniques can provide more complete explanations because the goal of global techniques is to provide an explanation that generalizes to any data instance.

Studying individual neurons versus distributed representations

Studying neurons individually without considering their interactions provides incomplete explanations. Explanation methods that focus on individual neurons [47, 50, 51, 54, 55, 5, 49, 6] typically only provide interpretable explanations for neurons that are highly selective of some concept. However, only a small fraction of individual neurons are highly selective of concepts [5, 54]. Furthermore, highly concept-selective neurons can be ablated without removing a model's ability to recognize that concept, meaning neurons that are not highly selective still play an important role in network function [85, 4, 42]. Thus, techniques that study neurons individually provide less complete explanations of DNNs because they do not consider neuron interactions and often disregard neurons that are not individually interpretable. To gain a global understanding of a DNN, every neuron must be studied and understood which can be incredibly time consuming given that popular DNNs have millions of neurons. Studying the preferred stimulus of individual neurons alone is not sufficient, and sometimes misleading, for understanding deep representations.

A better approach is to study distributed representations by considering combinations of neurons or vectors in activation space to produce explanations [55, 19, 61, 20, 4, 41, 32, 11]. Both biological and artificial neural networks have distributed representations meaning that multiple neurons fire together to represent concepts [19, 20, 4, 41]. To provide complete

explanations, it is imperative for XAI methods to consider how neurons are interacting rather than study them in isolation.

Supervised versus unsupervised methods

“Supervised” explanation methods attempt to identify **how a set of user-defined concepts is represented** in a DNN [5, 19, 49, 36, 61]. The supervision comes from the fact that these methods search for a pre-defined set of concepts. These supervised explanations are highly interpretable because user-defined concepts are all semantically meaningful. Since sets of user-defined concepts are unlikely to encompass the full set of concepts represented by a DNN, it is difficult to assess the completeness of these explanations [80]. Supervised explanation methods are difficult to apply to novel image applications because large labelled sets of user defined concepts are expensive to collect.

Unsupervised explanation methods attempt to **identify the concepts a DNN has learned to represent** [11, 22, 1, 34, 78, 40, 68, 32]. Unsupervised explanation methods are likely to produce more complete explanations than supervised methods because their explanations are not biased towards a set of user-defined concepts. Instead, the set of concepts identified by unsupervised methods depends only on the model’s learned representation.

2.2.2 Visualizing DNN activations in input space

To understand DNN representations, it is useful to be able to map a point in activation space back to input space where it can be visualized. Since traditional DNNs are non-invertible, this mapping between activations and inputs must be approximated somehow. The simplest method to visualize a point in activation space back in input space is to display a data instance that, when forward propagated, produces an activation similar to the point in activation space being visualized. Since the visualizations are simply data instances, they are interpretable. However only a narrow attribute of the data instance could be responsible for the proximity to the point being visualized. Displaying data instances that yield similar activations to the activation being visualized is both simple and interpretable.

Another approach is to optimize an image parameterization to maximize the activation of a particular neuron. This is often referred to as activation maximization. Simonyan *et al.* [70] and Mahendran *et al.* [47] optimized image parameterizations to maximize the activation of class output neurons. Zeiler *et al.* [83] used a deconvolutional network to approximately invert activations back to input space. Olah *et al.* [54] optimized image

parameterizations to maximize the activation of individual neurons, maximize the activation of multiple neurons, or approximately invert a target activation vector in input space. Methods based on optimizing an input parameterization to produce a desired activation are often uninterpretable [7] because they produce unnatural-looking visualizations with severe artifacts [70, 82, 47, 54].

To invert DNN activations back to input space, training a generative model yields the highest quality visualizations. Nguyen *et al.* [51] use a generative adversarial network to approximately invert CNN activations, producing visualizations resembling natural images. Rombach *et al.* [61] use a pipeline involving an invertible neural network and an autoencoder to approximately invert CNN activations back to input space, resulting in high quality, natural-looking visualizations. The main drawback of using generative models to invert activations is the large computational cost of training generative models. This cost can be prohibitive for many users who lack access to plentiful graphical processing unit (GPU) resources. Nonetheless, visualization techniques involving generative models [51, 61] produce the most interpretable visualizations.

2.2.3 Supervised explainability methods

Supervised explainability methods rely on a set of user-defined concepts to provide explanations. This can make the explanations more interpretable because users can already understand the set of concepts that they defined.

Bau *et al.* proposed network dissection [5], a technique that quantifies the alignment of individual neurons with user-defined concepts by the intersection over union (IoU) between the region containing a concept and the regions that elicit high activations from the neuron. If a neuron tends to activate highly in the presence of a particular concept, then that neuron is labelled as a detector of that concept. Network dissection relies on the Broden dataset [5] which contains 63,305 images with labelled concept segmentations with 1197 concepts in total. For AlexNet, the authors found that only a small fraction of neurons had high enough IoU with concepts to be labelled as detectors of those concepts.

Fong *et al.* [19] proposed net2vec, which extends network dissection to quantify the alignment between linear combinations of neurons. They demonstrated that linear combinations of filters are more aligned (much higher IoU) with concepts than individual neurons, suggesting that concepts are represented in a distributed manner within a layer.

Mu *et al.* [49] also extended network dissection to quantify the alignment between individual neurons, and logical compositions of user-defined concepts. For example, a neuron could be labelled as a detector of boats and water that is not green.

Testing with concept activation vectors (CAVs) [36] bears some similarity with net2vec [19] because it also involves searching for the linear combination of neurons whose activations align with the presence of a specified concept. Obtaining a CAV involves training a linear classifier to discriminate activations from inputs containing a specific concept, with activations from random counterexamples, then taking the vector orthogonal to the decision hyperplane as the CAV. Activations can be projected onto CAVs and their corresponding inputs can be visualized.

Aubry *et al.* [3] used a computer generated (CG) imagery system with image parameters such as object style, 3D viewpoint, color, and scene lighting configuration. To study how a parameter is represented in a CNN, they generated a set of CG images while varying the parameter, fit principal component analysis to the activations, and plotted the input images along the first few principal components to show how different layers encode the change in the input parameter. In this thesis PCA is also fit to CNN activations, however, this thesis does not control the set of concepts being varied in the input in any way.

Lenc *et al.* [45] also vary predefined input parameters but then perform regression to predict the value of input parameters given activations.

Esser *et al.* [18] and Rombach *et al.* [61] learn an invertible non-linear mapping between pretrained CNN activations and a set of predefined input parameters.

The completeness of the explanations from these supervised methods depends on how suitable the set of user-defined concepts is for explaining the particular combination of model, weights, input data, and task. Yeh *et al.* [80] mathematically defined the completeness of a set of concepts, then searched for the set of concepts that maximizes their completeness metric. While supervised methods provide interpretable explanations, it is very unlikely that any set of user-defined concepts is sufficient for providing a complete explanation of a network [80].

2.2.4 Unsupervised explainability methods

Unsupervised methods try to uncover the concepts that the model is representing. In contrast to supervised methods, unsupervised methods do not rely on a set of user-defined concepts to provide an explanation. Several works have applied unsupervised techniques to find structure in a DNN layer’s activations.

Some works [11, 34] sample activations from pretrained CNNs and then project the high-dimensional activations onto a two-dimensional manifold using UMAP [11, 48] and t-SNE [34, 76]. Points on the two-dimensional manifold can be sampled and visualized

either through optimization [11, 54] or by displaying the input data corresponding to each point [34]. By visualizing the embedding distances between visual concepts, unsupervised methods help us understand deep representations. However, by projecting high dimensional activation spaces down to two dimensions, much information is lost. For layer mixed4c in InceptionV1 [73], Carter *et al.* [11] manually found three paths along the two-dimensional manifold corresponding to a change in a stimulus attribute e.g. number of fruit, number of people, and blur level of foliage. These paths are tuning dimensions because they correspond to a change in a stimulus attribute. This result shows that it is possible to find semantically meaningful tuning dimensions based on a dataset of activations.

Another approach to finding structure in DNN activations is to factorize CNN activations with non-negative matrix factorization (NMF) [12, 55] to identify linear combinations of neurons which may correspond to different concepts, then visualize each linear combination of neurons using activation maximization [54]. Alammar [1] fits NMF to fully-connected layer activations from transformers to identify linear combinations of neurons which may correspond to different concepts. However, the number of components is typically chosen to be much less than the number of neurons, resulting in less complete explanations.

Bolei *et al.* [78] and Ghorbani *et al.* [22] fit clustering algorithms to CNN activations from an entire training set to obtain cluster centers corresponding to the concepts a layer represents. Bolani *et al.* [78] display image patches which correspond to cluster centres. Ghorbani *et al.* [22] find activation vectors for each cluster centre, then display image super-pixels projected onto each activation vector.

Lang *et al.* [40] train a StyleGAN to learn semantically meaningful image attributes which affect a particular classifier’s outputs. By varying the StyleGAN activations along the semantically meaningful image attributes, they provide a counterfactual explanation which shows how manipulating an image attribute will affect a classifier’s decision.

2.2.5 Manual studies of DNN representations

There has been some work on understanding deep representations by manually examining deep networks. Cammarata *et al.* [10] manually identified that the angular orientation of curves in the input is a tuning dimension for curve detector neurons in multiple InceptionV1 [73] layers. The activation of multiple neurons can be explained by a single stimulus attribute. This result demonstrates the value of tuning dimensions for producing decipherable explanations of the behaviour of groups of neurons. Olah *et al.* [53] published a study of individual neurons in the early layers of InceptionV1 [73] where they grouped neurons by their function.

2.2.6 Understanding the activation space of generative models

Compared to discriminative networks, visualizing how variance in a generative model’s activation space relates to image space is simple: vary activations, then forward pass to obtain a generated image. Several works [68, 32, 33, 77, 27, 6] identify directions to edit activations which correspond to changes in semantically meaningful attributes of the generated images. A method called GANSpace [32] takes an unsupervised approach by fitting PCA to a set of sampled activations, then vary activations along each principal component. Principal components define remarkably disentangled edit directions for generative models such as StyleGAN2 [35] and BigGAN [8].

2.2.7 Human XAI studies

Quantifying the interpretability of explanations is inherently subjective. Conducting a user study is the most rigorous method to measure explanation interpretability [41, 16]. Several DNN XAI papers have conducted user studies to evaluate the interpretability of their explanations [2, 25, 67, 7].

Many user studies ask participants to perform the task of forward simulation [2, 25, 67]. In a forward simulation task, users are first presented with explanations of a model. Then given a previously unseen input, they must predict the model’s output using the information from the explanations they were presented [16]. To perform well on a forward simulation task, explanations must generalize to many data instances.

Another common type of user study is to present a N-alternative forced choice task [7, 32, 40, 67]. In the context of XAI method, two explanations are typically presented, one being the true explanation and the other being a random uninformative explanation. Users are then forced to choose which explanation is the most informative. Borowski *et al.* [7] conducted a forced choice user study and found natural image explanations to be more interpretable than optimized feature visualizations [54].

2.2.8 Connection to neuroscience

Computational neuroscientists are faced with a similar task as XAI researchers: they aim to develop techniques to explain the function of biological neural networks in the brain. The cerebral cortex has a number of similarities with DNNs, such as layers of simple units and adaptive connection weights. Biological neural networks found in the cerebral cortex are more complex than DNNs; yet neuroscientists have developed several methods

to understand aspects of cortical function [81, 38, 74]. These methods are time consuming and expensive to apply because they involve measuring brain activity with techniques such as EEG, fMRI, or electrodes recording individual neurons. The cost of measuring brain activity limits the complexity of task and duration of brain activity that can be studied. Fortunately, it is fast and inexpensive to measure the activations of artificial neurons in DNNs. This enables deep learning researchers to easily measure millions of activations from DNNs as they perform highly complex tasks.

A method used in both neuroscience and deep learning is activation maximization; finding the input that maximizes the activation of an individual neuron, or a collection of neurons [70, 74]. The input found through activation maximization is also called the preferred stimuli. If the preferred stimuli of several neurons are found to be related, they can be used to identify a tuning dimension for a group of neurons.

Tuning dimensions are foundational methods for understanding groups of biological neurons because they can provide a decipherable explanation of what groups of neurons respond to [21, 30, 52]. A tuning dimension is a stimulus attribute that accounts for much of the activation variance of a group of neurons. For example, the activations of a neurons in a cat’s visual cortex were maximized when a bar of light was presented at particular angles (eg. 30, 45, 60 degrees) [30]. The angle of the bar of light in this case is the tuning dimension of the population of cat neurons. Since data collection in biological neurons is expensive and time consuming, the tuning dimensions for a group of neurons usually become clear over decades following painstaking trial-and-error. Tuning dimensions are typically defined by non-linear paths through activation space.

As Cunningham *et al.* [13] discuss, applying dimensionality reduction to biological neural network activations has been done many times before. Since recording activations from biological neuron populations is time-limited, expensive, and invasive, experiments are typically much smaller scale than DNN experiments and usually attempt to study a few specific functions of a neural population during a simple task.

When PCA is fit in biological neural activations, the variance in the activations of a high-dimensional neuron population may be explained with a few components [20, 65, 72]. Stopfer *et al.* [72] fit both PCA and locally linear embedding (LLE) to locust brain activations and show that non-linear LLE dimensions disentangle recognition of specific odors. Russo *et al.* [65] recorded activations from the primary motor cortex (M1) and supplementary motor area (SMA) of monkeys while the monkeys rotated a hand crank. They fit PCA to the high-dimensional recorded activations, then in the embedding space of the first three principal components, they showed that the activation trajectory over time corresponded to a cyclical, non-linear path.

Gallego *et al.* [20] recorded activations from the primary motor cortex (M1) of monkeys while they performed a simple task with a joystick, fit PCA, projected activation trajectories onto the first three principal components, and demonstrated a non-linear path was contained within this embedding. They hypothesize that the variance of high-dimensional activations can be explained by a lower-dimensional manifold in activation space. They also suggest that studying individual neurons can divert one from understanding neural representations since neurons can be involved in multiple neural modes.

A common theme amongst these works [13, 20, 65, 72] is that PCA provides a linear basis for an inherently non-linear phenomenon in activation space. Gallego *et al.* [20] pointed out that the intrinsic dimension of a non-linear path in activation space is less than the path’s PCA embedding dimension. PCA constructs a basis by identifying orthogonal linear combinations of neurons that maximize the projected variance along each basis. The orthogonality constraint of PCA attempts to minimize any linear correlations between PCs, but PCs can still have nonlinear relationships with each other. Stopfer *et al.* [72], Russo *et al.* [65], and Gallego *et al.* [20] each identified nonlinear paths through the embedding space of the top principal components that corresponded to a change in a meaningful stimulus attribute (a tuning dimension). This suggests that biological tuning dimensions are encoded by non-linear paths through activation – mirroring the evidence of non-linear paths corresponding to tuning dimensions in DNN activations [10, 11].

Chapter 3

Prologue to first article

3.1 Article details

Identifying and interpreting tuning dimensions in deep networks. In *Shared Visual Representations in Human & Machine Intelligence NeurIPS Workshop, 2020*. Nolan S. Dey, J. Eric Taylor, Bryan P. Tripp, Alexander Wong, Graham W. Taylor

3.1.1 Personal contributions

The idea to study the tuning dimensions of DNNs came jointly from Bryan P. Tripp, Alexander Wong, and Graham W. Taylor. As the first author, I was responsible for the programming, experimentation, and writing of the majority of the manuscript, under the supervision of my four co-authors.

3.2 Context

In neuroscience, one of the foundational abstractions for understanding related groups of biological neurons is the concept of tuning dimensions [21, 30, 52]. A population or group of neurons produces activations based on a stimulus being presented. A tuning dimension is a stimulus attribute that explains the activations of a group of neurons. For example, Figure 3.1 shows a population of 10 neurons that each activate highly when a stimulus

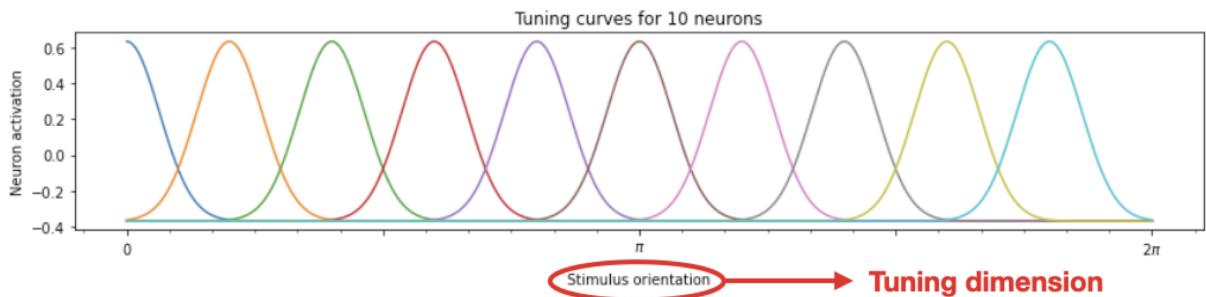


Figure 3.1: Toy example of tuning curves for a population of neurons. The tuning dimension of this population is the stimulus orientation.

is oriented at a certain angle. The tuning dimension in this case is the orientation of the stimulus because it can explain the activations of the group of neurons.

While some researchers have attempted to manually identify tuning dimensions in DNNs [10, 11] prior to this article work, there was no automatic way to identify them. In a NeurIPS 2020 workshop paper, a framework is presented for automatically identifying an analogue of tuning dimensions in DNNs. It is shown that the PCA and ICA bases for activation space automatically identify the expected tuning dimensions for early vision layers in InceptionV1 and a synthetic Gabor bank.

However, the method presented in this article has one major limitation. Based on the visualization techniques used, it is very hard to interpret layers without expected tuning dimensions. This limited the paper to only studying early vision layers because deeper layers could not be studied with confidence. In the second article this limitation is addressed.

3.3 Contributions

This work contributes an unsupervised framework for identifying and interpreting “tuning dimensions” in DNNs. The method correctly identifies the tuning dimensions of a synthetic Gabor filter bank, and those of the first two layers of InceptionV1 trained on ImageNet.

Chapter 4

Identifying and interpreting tuning dimensions in deep networks

4.1 Introduction

In neuroscience, tuning dimensions are foundational methods for understanding groups of biological neurons because they can provide a decipherable explanation of what groups of neurons respond to [21, 30, 52]. A tuning dimension is a stimulus attribute that accounts for much of the activation variance of a group of neurons. Since data collection in biological neurons is expensive and time consuming, the tuning dimensions for a group of neurons usually become clear over decades following painstaking trial-and-error.

In deep learning, these data collection limitations do not exist because we can easily measure neural activations for very large and diverse datasets. Some works have manually found an analogue to tuning dimensions in deep neural networks (DNNs) [10, 11]; however, there is currently no way to automatically identify them. In this paper, we define such a method as follows. First, we acquire a dataset of activations. Then, we apply unsupervised dimensionality reduction techniques to identify a set of basis vectors that explain that activation space’s variance. Finally, we visualize points [54] along each basis vector to produce the tuning dimension. Our method identifies the tuning dimensions of a synthetic Gabor filter bank and our hypothesized tuning dimensions of the first two layers of InceptionV1 [73].

4.2 Related work

A popular approach to explaining the function of neurons in convolutional neural networks (CNNs) is to visualize their preferred input. These methods are compelling because the resultant images are highly interpretable and because the explanations can be generated automatically. A neuron’s preferred stimulus can be visualized by optimizing an input image to maximize the activation of a particular neuron [47, 50, 51, 54, 55]. Some neurons show strong activation in response to a diverse range of stimuli [50, 54]. Furthermore, neurons rarely fire in isolation — multiple neurons often fire together to represent concepts [55]. Together, these findings suggest that studying the preferred stimulus of individual neurons alone is not sufficient, and sometimes misleading, for understanding deep representations.

To gain a more complete understanding of deep representations, we should strive to understand the activation space of each layer. Once trained, layers in deep neural networks transform the activation space of the previous layer in a way that minimizes some loss function. A point in activation space \mathbf{a} can be visualized by optimizing an input image I to approximately reproduce \mathbf{a} when I is forward propagated through the network [54]. It is difficult to study activation space because it is high-dimensional, even with a visualization method. [11, 34] project the high-dimensional activation space to a two-dimensional manifold while attempting to preserve high-dimensional distances. Then, points on the two-dimensional manifold can be sampled and visualized either through optimization [11] or by displaying the input data corresponding to each point [34]. These methods help us understand deep representations by visualizing the embedding distances between visual concepts. However, by compressing high dimensional activation spaces to two dimensions, a great deal of information is lost. When identifying tuning dimensions, we need not constrain ourselves to compressing activation space to two or three dimensions because a layer can have more than three tuning dimensions.

There has also been some work on understanding deep representations by manually identifying tuning dimensions in deep networks. For layer mixed4c in InceptionV1 [73], Carter *et al* [11] manually found three paths along the two-dimensional manifold corresponding to a change in a stimulus attribute e.g. number of fruit, number of people, and blur level of foliage. These paths are tuning dimensions because they correspond to a change in a stimulus attribute. This result shows that it is possible to find semantically meaningful tuning dimensions based on a dataset of activations. Cammarata *et al* [10] manually found that the angular orientation of curves in the input is a tuning dimension for curve detector neurons in multiple InceptionV1 [73] layers. In other words, the activation of multiple neurons can be explained by a single stimulus attribute. This result demonstrates the value of tuning dimensions for producing decipherable explanations of the behaviour of groups of

neurons. Though this manual process of identifying tuning dimensions bears some similarity to neuroscience, it is a time consuming process. We are unaware of any existing methods for automatically identifying tuning dimensions in DNNs.

4.3 Method

In this paper we investigate layers of InceptionV1 [73] pretrained on ImageNet [14]. Prior works have also studied this combination [10, 11, 54, 55, 53]. ImageNet is a large, diverse sample of natural images and InceptionV1 was previously state-of-the-art for the ImageNet classification task [64].

4.3.1 Collecting activations

We first obtain a representative sample of a layer’s activations. This is the same method of collecting activations used by Carter *et al* [11]. We forward propagate N randomly chosen ImageNet [14] images through InceptionV1 up to a specified layer to obtain an $(N \times h \times w \times c)$ matrix of activations, where h and w are the spatial height and width, and c is the number of channels. Following [11], we found $N = 1 \text{ M}$ to be sufficient although it is likely that a much lower N would also be sufficient. If we collected activations from the center spatial position, we would be biasing our collection towards input features typically found at the center of images. To obtain a representative sample of activation space while also respecting memory constraints, we choose a random spatial index in the activation map (padded by one spatial position to avoid boundary artifacts) to collect an activation vector (\mathbb{R}^c) for each image to obtain activations $A \in \mathbb{R}^{N \times c}$.

4.3.2 Identifying tuning dimensions

We apply a dimensionality reduction technique to \mathbf{a} to obtain a set of lower dimensional transformed activations $A' \in \mathbb{R}^{N \times n}$ where n is the number of reduced dimensions. We treat each of the n dimensions of A' as tuning dimensions. We compare the scikit-learn [58] implementations of several dimensionality reduction techniques in Section 4.4.1: principal component analysis (PCA), independent component analysis (ICA) [31], non-negative matrix factorization (NMF) [12], and locally linear embedding (LLE) [62].

4.3.3 Visualizing points along each tuning dimension

To visualize a tuning dimension, we sample m points (\mathbb{R}^n) along the tuning dimension to obtain an $(m \times n)$ matrix of sampled points in transformed activation space. Then, we apply an inverse transform to obtain an $(m \times c)$ matrix of sampled points in untransformed activation space. We found that uniformly sampling m points between the observed minimum and maximum values along each tuning dimension produced the most informative visualizations. Sampling methods are compared in Section A.1.

Let $\mathbf{a}' \in \mathbb{R}^{h \times w \times c}$ be the activation obtained from forward propagating an image parameterization I up to a specified layer. Following Olah *et al* [54], we visualize a sampled point in activation space $\mathbf{a} \in \mathbb{R}^c$ by optimizing an image parameterization I to maximize the mean cosine similarity between \mathbf{a} and \mathbf{a}' as follows:

$$I^* = \arg \max_I \left(\frac{1}{wh} \sum_i^w \sum_j^h \frac{\mathbf{a} \cdot \mathbf{a}'_{ij}}{\|\mathbf{a}\| \|\mathbf{a}'_{ij}\|} \right) \quad (4.1)$$

This optimization results in an image I^* which approximately reproduces \mathbf{a} at every spatial location in the activation map \mathbf{a}' of the intended layer. We also tried optimizing an image I to reproduce \mathbf{a} at a single spatial location in \mathbf{a}' but this resulted in less interpretable visualizations.

The torch-lucent library was used for visualization.¹ Following Olah *et al* [54], we applied small random affine transformations to I each optimization step, and the image parameterization I we used was the Fourier basis with all frequencies scaled to have the same energy. When we used a pixel-based image parameterization I instead, the resulting visualizations I^* resembled noise. However, prior work suggests that if we were visualizing the activations of a network that was trained with an adversarial robustness objective, optimizing a pixel-based image parameterization I should yield more interpretable visualizations [17].

4.3.4 Interpreting tuning dimension visualizations

A tuning dimension is a stimulus attribute that accounts for much of the activation variance of a group of neurons. While studying a tuning dimension visualization, a change in a stimulus attribute along the dimension is expected. The interpretation of the changing

¹<https://github.com/greentfrapp/lucent>

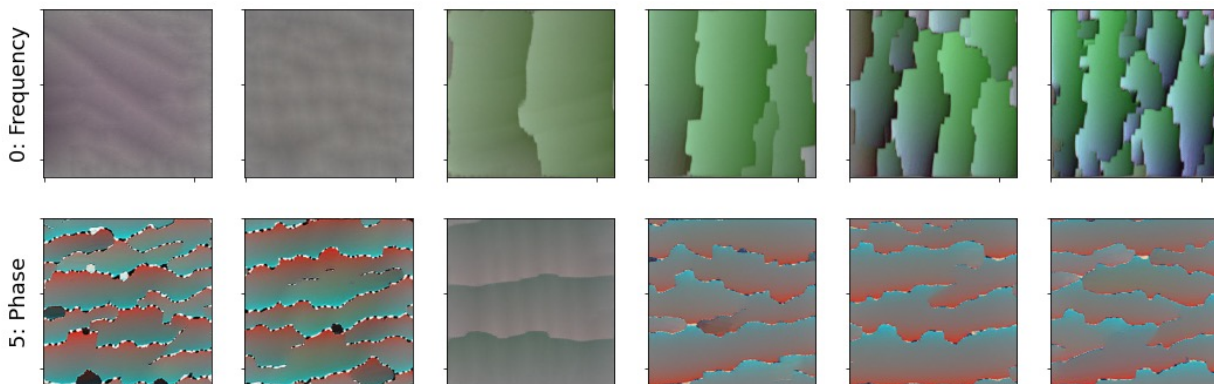


Figure 4.1: Tuning dimension visualizations of the first layer of InceptionV1 (conv2d0). Top: Visualized points along first ICA component show increasing frequency of vertically oriented edges. Bottom: Visualized points along sixth ICA component show dark/light edges switching to light/dark edges which is associated with a change in Gabor filter phase. 32 points were visualized for each tuning dimension but to improve readability, the most informative 6 are shown. Best viewed in color.

stimulus attribute(s) is a subjective process. We are more likely to notice changes in a hypothesized stimulus attribute, which introduces the danger of confirmation bias. To provide transparency, our interpretations are included in the Appendix sections [A.3](#) and [A.4](#).

Points in A' obtained from PCA or ICA can take both positive and negative values. When examining points along a dimension of A' obtained from PCA or ICA, the points at the positive and negative ends represent opposite concepts, while the points near zero represent the mean collected activation vector. Figure 4.1 shows visualizations of two tuning dimensions identified by ICA for conv2d0, the first layer of InceptionV1 [73]. Points in A' obtained from NMF can only take positive values.

4.4 Results

In Section 4.4.1, we compare dimensionality reduction models and demonstrate that our method can correctly identify the tuning dimensions of a synthetic Gabor bank. In Section 4.4.2, we study the tuning dimensions of the first two layers of InceptionV1 [73] and show that our method can identify a set of hypothesized tuning dimensions. We found it best to

study these visualizations through an interactive interface.²

4.4.1 Identifying tuning dimensions of a synthetic Gabor bank

To compare dimensionality reduction methods for identifying tuning dimensions, we need a method to validate the recovery of known tuning dimensions. To ensure the correct tuning dimensions are known *a priori*, we construct a bank of Gabor filters with tuning dimensions of frequency, angle, and phase. Both the human visual system and trained CNNs contain Gabor-like filters in early layers [53, 56]. Frequency, angle, and phase affect the filter’s preferred edge frequency, edge angle, and dark/light edge orientation respectively. Since conv2d0, the first layer of InceptionV1 [73], has 28 7×7 Gabor filters [53] applied with a stride of 2, we used the same number of filters, kernel size, and stride for the Gabor filter bank. More details regarding the Gabor filter bank’s construction are included in the Appendix section A.2.

Table 4.1: Comparison of identified Gabor bank tuning dimensions. PCA-3 denotes PCA with 3 components.

Tuning Dimension	PCA-3	PCA-6	ICA-3	ICA-6	NMF-3	NMF-6	LLE-3	LLE-6
Frequency	✓	✓	✓	✓	✗	✗	✗	✗
Angle	✗	✗	✓	✓	✗	✗	✗	✗
Phase	✓	✓	✓	✓	✗	✗	✗	✗

We collected activations and applied standard unsupervised learning techniques to identify the tuning dimensions: PCA, ICA [31], NMF [12], and LLE [62]. Table 4.1 shows that only ICA correctly identified all of the tuning dimensions, PCA failed to identify angle, and both NMF and LLE failed to identify any of the tuning dimensions. PCA may have failed to identify the angle tuning dimension because each component is constrained to be orthogonal whereas ICA may have succeeded because it does not have this constraint. The visualizations of points along each NMF tuning dimension did not have any meaningful variation. This could be due to the NMF constraint that points along each tuning dimension be non-negative, meaning no meaningful variation was observed in the positive region of the NMF tuning dimensions. In the positive regions along PCA and ICA tuning dimensions, it is also rare for meaningful variations to be observed. However, the meaningful variation along PCA and ICA tuning dimensions is observed due to the transition from the negative

²Link to interactive demo: <https://tinyurl.com/tuning-dimensions-svrhm>

to positive regions. Visualized points along each LLE tuning dimension were very different but had no apparent pattern between them. We continue to use both PCA and ICA because they identified most of the Gabor filter bank tuning dimensions. We wanted a method that is not sensitive to hyperparameters so we did not try deep learning-based dimensionality reduction techniques.

4.4.2 Identifying tuning dimensions of InceptionV1 layers

There is no quantitative metric to evaluate the validity of tuning dimensions. Fortunately, Olah *et al* [53] published a study of individual neurons in the early layers of InceptionV1 which could be used as a reference to hypothesize the ground-truth tuning dimensions in this particular architecture. When evaluating different variations of our method, we compared our results to the tuning dimensions we had hypothesized. In other words, we ask whether our method for automatic discovery of tuning dimensions converges on the same dimensions manually identified by other researchers.

conv2d0

Olah *et al.* [53] manually categorized the `conv2d0` neurons into Gabor, color contrast, and uncategorized families. We hypothesized that the Gabor neurons would have the same tuning dimensions as our synthetic Gabor bank: frequency, angle, and phase (dark/light orientation of edges). Based on the color contrast neurons, we expected to see green/purple, orange/blue, green/red, and blue/red color contrasts. We did not hypothesize any tuning dimensions for the uncategorized units. After fitting ICA with 16 components and visualizing the tuning dimensions, all of the hypothesized tuning dimensions were identified (Table 4.2). Each of the visualizations that we interpreted as one of the hypothesized tuning dimensions is included in Section A.4.

conv2d1

Olah *et al.* [53] manually categorized the `conv2d1` neurons into low frequency, Gabor, color contrast, complex Gabor, multicolor, color, hatch, and uncategorized families. For the Gabor and color contrast neurons we hypothesized the same tuning dimensions as `conv2d0`. We also expected the complex Gabor filters to have frequency and angular orientation tuning. We did not hypothesize any tuning dimensions for the hatch and uncategorized neurons because the hatch category consisted of a single neuron, and the 3 uncategorized neurons

Table 4.2: Comparison of PCA and ICA for identifying tuning dimensions in the InceptionV1 layers `conv2d0` and `conv2d1`. PCA-16 denotes PCA with 16 components.

Hypothesized Tuning Dimension	<code>conv2d0</code>		<code>conv2d1</code>	
	PCA-16	ICA-16	PCA-16	ICA-16
Gabor frequency	✗	✓	✓	✓
Gabor angle	✓	✓	✓	✓
Gabor phase	✓	✓	✓	✓
Green/purple contrast	✓	✓	✓	✓
Orange/blue contrast	✓	✓	✓	✓
Green/red contrast	✗	✓	✗	✓
Blue/red contrast	✓	✓	✓	✓

had no apparent relationship. ICA with 16 components identified all of the hypothesized tuning dimensions (Table 4.2). Each of the visualizations that we interpreted as one of the hypothesized tuning dimensions is included in Section A.4.

4.5 Conclusions and future work

We propose the first general unsupervised method for identifying and visualizing the tuning dimensions of a deep neural network layer. We show the method correctly identifies the tuning dimensions of a synthetic Gabor filter bank and hypothesized tuning dimensions of the first two layers of InceptionV1. Moving forward, we have identified a number of areas for future work.

A key area of future work will be studying the higher layers of InceptionV1 in more detail. It takes approximately 2 minutes to manually interpret each tuning dimension. The layers after `conv2d1` have many neurons (≥ 192), making it more time consuming to study these layers. The higher layers of InceptionV1 [73] also contain more diverse and complex neurons, making the tuning dimensions of each layer unclear. Without clear hypotheses for a layer’s tuning dimensions, this introduces a great deal of uncertainty into the qualitative evaluation of the tuning dimensions.

Our method is not specific to a particular data instance, making it a global explainability method. Once our method is used to label the tuning dimensions of a layer, the labels may be used in a “what-if” tool [79] to provide instance-specific explanations. We may also use

an external dataset such as Broden [5] to quantitatively associate each tuning dimension with specific visual concepts.

Finally, our method may be extended to identify tuning dimensions in reinforcement learning, natural language processing, and graph learning models provided a method exists to visualize an activation \mathbf{a} by optimizing an input parameterization to reproduce \mathbf{a} . Ponce *et al* [59] proposed a method for optimizing the preferred stimuli I^* of individual biological neurons. Their method could be extended to optimize stimuli I^* which reproduce a particular activation vector \mathbf{a} . This extension could allow our method to be applied to biological neural networks.

Chapter 5

Prologue to second article

5.1 Article details

The surprising effectiveness of PCA for understanding CNN representations. *Submitted to NeurIPS 2021 for review.* Nolan S. Dey, J. Eric Taylor, Bryan P. Tripp, Alexander Wong, Graham W. Taylor

5.1.1 Personal contributions

As the first author, I was responsible for the programming, computational experimentation, and the writing of the majority of the manuscript, under the supervision of my four co-authors. In addition to providing supervision, the second author J. Eric Taylor was responsible for the user study. He conceived the experimental design, applied for the IRB, recruited participants, conducted the study, and analyzed the results.

5.2 Context

As mentioned in Section 3.2, the method in the first article had a major limitation: it was difficult to interpret the synthetic visualizations that were produced, especially for deeper layers.

In the first article, the authors reported their own interpretations of the synthetic visualizations for early vision layers. To test whether other computer vision researchers

shared the authors’ interpretations, a user study was conducted with 8 participants. For tuning dimensions in the first two layers it is known that some of them correspond to the Gabor filter parameters of frequency, angle, and phase. Users were presented with a visualization and asked them if they saw one or more of the 3 Gabor filter parameters. The task presented to each participant in the user study are shown in Figure 5.1.

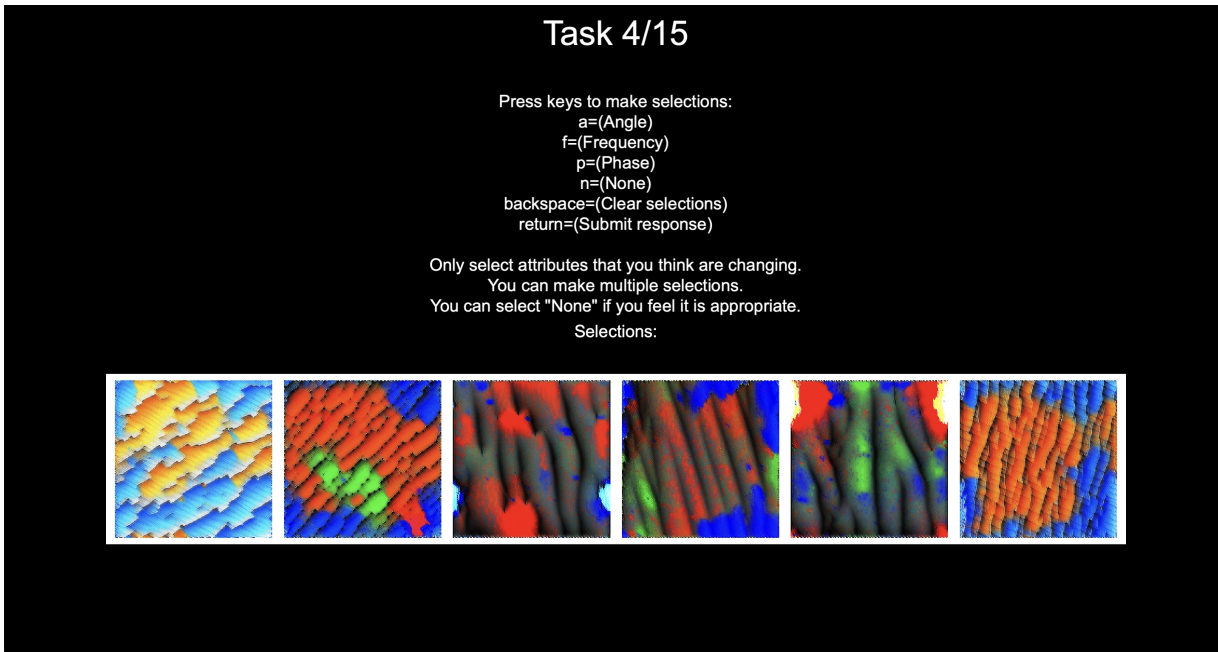


Figure 5.1: Task presented to participants of the user study.

Krippendorff’s alpha (α) is a metric used to quantify inter-rater reliability. In other words, if multiple participants respond to the same prompt, α quantifies the agreement between the participants’ responses. $\alpha = 1$ means there is perfect agreement between participants, $\alpha = 0$ means participants’ responses are statistically unrelated, and $\alpha < 0$ is worse than random chance and suggests there is a systematic disagreement.

Most participants agreed on visualizations corresponding to a change in angle, but their agreement approached random chance for frequency ($\alpha = 0.35$) and phase ($\alpha = -0.0008$). This user study shows that even for simple tuning dimensions, synthetic visualizations alone do not provide interpretable explanations, even to computer vision experts.

This user study made it clear that a more interpretable visualization technique was needed. The second article builds upon the first article by developing an additional visualization technique that greatly improves explanation interpretability, enabling the

interpretation explanations of deeper layers with more confidence. It was also found that studying representations before the ReLU non-linearity is applied greatly improves the completeness and interpretability of the explanations.

Many popular DNN explanation methods make choices that increase interpretability at the expense of completeness. There are 3 such choices highlighted in this work: relying on instance-specific explanations, studying neurons individually, and relying on a set of user-defined concepts for explanations. To provide a more complete explanation of high-dimensional and distributed DNN activations, the principal components (PCs) of DNN layer activations are visualized.

The PCs found are analogous to tuning dimensions, however, they do not accurately capture the true tuning dimensions because true tuning dimensions are non-linear. The second article has a change in vocabulary compared to the first article: rather than describing PCs in activation space as tuning dimensions, they are described as basis vectors that can help one study the relationship between activation variance and input variance.

By combining ablation experiments and studies with human participants, it was found that the most important principal components formed a more complete and interpretable basis than individual neurons. Much of the activation variance may be understood by studying relatively few high-variance PCs, as opposed to studying every neuron. These PCs also strongly affect network function, and are highly interpretable. In contrast, while some neurons are highly interpretable, many of these have little individual impact on network function. As a result, the PC basis for DNN activations is both more interpretable and complete. New insights into deep representations may be gained by visualizing PCs in activation space.

5.3 Contributions

For AlexNet [39], this article shows that the principal components in activation space capture semantically meaningful concepts at each layer. Furthermore this work demonstrate that the PCA basis offers a more complete explanation of activation space than the neuron basis. Through a user study this work also show that the top principal components are more interpretable than the most important neurons.

Chapter 6

The surprising effectiveness of PCA for understanding CNN representations

6.1 Introduction

Safety critical applications of machine learning require transparent models that experts can understand and trust. The lack of complete and interpretable explainability methods for discriminative deep neural networks (DNNs) is a major obstacle to the adoption of DNNs in settings such as healthcare and transportation. With explainability methods, there is a tradeoff between interpretability and completeness [23]. Completeness describes how thoroughly an explanation describes a model while interpretability describes how understandable an explanation is [23]. For example, explaining a network’s function by writing down an equation with millions of terms would be the most complete explanation but also the least interpretable. Conversely, saliency map explanations [70] are highly interpretable but offer very incomplete explanations of a DNN [37]. Many popular DNN explanation methods make choices that increase interpretability at the expense of completeness.

Studying neurons individually without considering their interactions provides incomplete explanations. Both biological and artificial neural networks have distributed representations; multiple neurons often fire together to represent concepts [55, 19, 61, 20, 4, 41, 32]. Explanation methods that focus on individual neurons [47, 50, 51, 54, 55, 5, 49, 6] typically only provide interpretable explanations for neurons that are highly selective of some concept. However, only a small fraction of individual neurons are highly selective of concepts [5, 54]. Furthermore, highly concept-selective neurons can be ablated without removing a model’s ability to recognize that concept, meaning neurons that are not highly selective

still play an important role in network function [4]. Thus, techniques that study neurons individually provide less complete explanations of DNNs because they do not consider neuron interactions and often disregard neurons that are not individually interpretable.

Whereas neuron-based explanations can be difficult to connect with concepts, other methods perform a supervised search to identify how a set of user-defined concepts is represented in a DNN [5, 19, 49, 36, 61]. These supervised explanations are highly interpretable because user-defined concepts are semantically meaningful. Since user-defined sets of concepts are unlikely to encompass the full set of concepts represented by a DNN, supervised explanation methods are unlikely to provide complete explanations. Furthermore, neuron responses often relate weakly to such concepts, and supervised explanation methods are difficult to apply to novel image applications, because large labelled sets of user defined concepts are expensive to collect.

Our goal is to understand the high-dimensional and distributed activation space of a layer, and thus understand how the layer represents input stimuli. We sample a large set of activations, then use unsupervised principal component analysis (PCA) to identify a set of orthogonal basis vectors, ordered by the amount of activation variance they capture. Then to understand the activation variance along a principal component (PC), we visualize points along the component so we can relate activation variance to input variance. We show that the often-overlooked PCA basis for AlexNet [39] pretrained on ImageNet [14] is surprisingly interpretable and has a number of useful properties. Firstly, much of a layer’s activation variance is explained by only studying a fraction of PCs, as opposed to studying every neuron. Through ablation experiments we found that PCs with large eigenvalues are generally more important for AlexNet’s performance than the most important neurons, and that PCs with small eigenvalues are generally less important for AlexNet’s performance than the least important neurons. Through a user study we also show that PCs with large eigenvalues tend to be interpretable and PCs with small eigenvalues are generally uninterpretable. Moreover, these components are more interpretable than visualizations of the most important neuron bases. The results show the most important PCs form a more complete and interpretable basis than the most important neurons.

6.2 Method

DNN activations are high-dimensional, distributed, and contain non-linear phenomena. To understand activation space and the neuron interactions within it, we sample the space then fit PCA to obtain a set of basis vectors for the activation space. We then visualize points

along each basis vector and interpret how activation variance relates to input variance. Figure 6.1 shows an overview.

Despite the existence of non-linear phenomena in activation space, PCA is surprisingly effective at identifying an interpretable and complete basis. We found that it was important to fit PCA to pre-ReLU activations, resulting in variance being explained by far fewer PCs than if we fit to post-ReLU activations (see Section 6.3.2). The application of ReLU may make linear methods like PCA less effective because it increases the non-linearity of activations by setting negative activations to zero. To increase interpretability, we also found it crucial to include natural image visualizations and visualize points in activation space using distance minimization rather than activation maximization objectives (see Section 6.2.3).

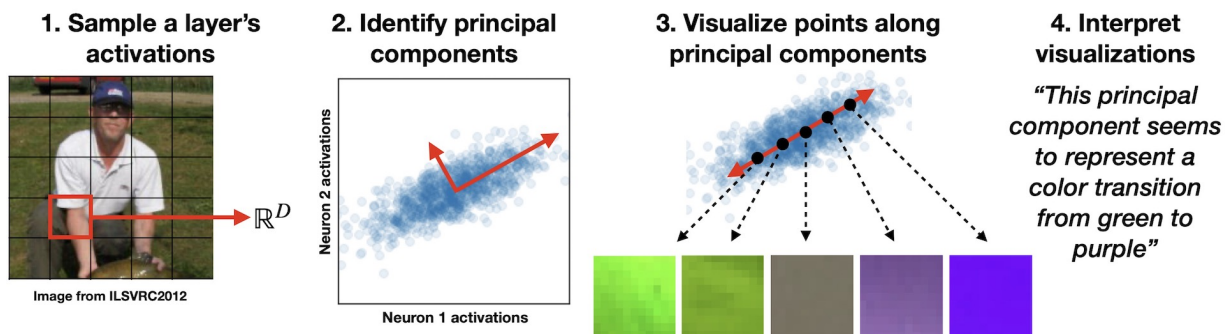


Figure 6.1: Overview of our method. We sample \mathbb{R}^D activations from a layer’s activation map (1), then identify the principal components of the layer’s activation space (2). Finally we visualize points along each principal component (3) and interpret the visualizations (4).

6.2.1 Sample a layer’s activations

Before fitting PCA to determine basis vectors, we first obtain a representative sample of a layer’s activations. When forward propagating an image through a DNN up to a specific fully connected layer, we store the full \mathbb{R}^D activation vector where D is the number of neurons. When forward propagating an image through a DNN up to a specific convolutional layer, we obtain $(H \times W \times D)$ matrix of activations, where H and W are the feature map height and width, and D is the number of channels. Since in a convolutional layer the same operation is applied at every spatial position, we can sample an activation vector (\mathbb{R}^D) at a single spatial position for each image to reduce the dimensionality of the layer’s activations and reduce memory requirements. However, if we sample activations from the

same spatial position for every image, we would bias our sampling towards input features typically found at the specific spatial position. To avoid biasing our sampling, we choose a random spatial position to sample for each image, avoiding boundary positions. We forward propagate every image in the training set through a DNN up to a specified layer, then sample a random spatial position to obtain an activation matrix $\mathbf{A} \in \mathbb{R}^{N \times D}$ where N is the length of the training set, and D is either the number of convolutional channels or the number of fully-connected neurons, depending on the layer. This is the same method of sampling activations used by [11].

6.2.2 Identify principal components

Activation space is high dimensional and there are many possible bases. The neuron basis, defined by axis-aligned unit vectors corresponding to each neuron, is the standard basis for the activation space. PCA is a simple and effective tool for understanding high-dimensional spaces. We apply PCA to \mathbf{A} to obtain a set of transformed activations $\mathbf{A}' \in \mathbb{R}^{N \times P}$ where P is the number of principal components (PCs). PCA finds a set of orthogonal basis vectors for \mathbb{R}^D activation space, ordered by the amount of activation variance they capture. If activation space has some covariance structure, then most of the activation variance can be explained with relatively few PCs. Significant activation covariance also implies neuron interactions are responsible for much of the activation variance. Previous works have also shown the PCA basis is useful for analyzing activations in both biological neural networks [20] and GANs [32]. We use the scikit-learn [58] PCA implementation for our experiments.

6.2.3 Visualize points along principal components

From PCA we obtain a transformed activation space \mathbb{R}^P . To understand the activation variance along each PC, we sample points along each PC (while zeroing out contributions from all other PCs) and visualize them in input space. Specifically, we sample m points (\mathbb{R}^P) along each PC (while zeroing out contributions from all other PCs) to obtain a $(m \times P)$ matrix of sampled points in transformed activation space. Then, we apply an inverse transform to obtain an $(m \times D)$ matrix of sampled points in untransformed activation space (\mathbb{R}^D). We found that uniformly sampling m points between the observed minimum and maximum values along each basis vector produced the most interpretable visualizations. Sampling methods are compared in Section B.3.

Each of the m points along each PC is visualized independently in input space using the three methods shown in Figure 6.2: synthetic receptive field (Synthetic-RF) [54], synthetic

full size (Synthetic-Full) [54], and natural receptive field (Natural-RF).

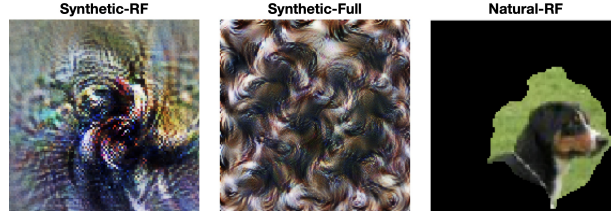


Figure 6.2: Examples of synthetic receptive field (Synthetic-RF) [54], synthetic full size (Synthetic-Full) [54], and natural receptive field (Natural-RF). Although Synthetic-Full has a higher resolution than Synthetic-RF and Natural-RF, each of the visualizations is scaled to be the same size.

When we visualize neurons or components, we use distance minimization rather than the more common activation maximization objective. Activation maximization involves producing a visualization that maximizes the activation along some neuron or some PC. The issue with activation maximization is that there are many degrees of freedom; to maximize neuron A, a visualization may also elicit large eigenvalue PCs from neuron B and C. This freedom makes studying distributed representations difficult because neurons or PCs cannot be studied independently of each other. Distance minimization involves producing a visualization that minimizes the distance between the visualization’s response in activation space, and some target point in activation space. This allows us to study how variance along a single basis vector in activation space relates to variance in input space.

Synthetic visualization

Following Olah et al. [54], we visualize a point in activation space $\mathbf{a} \in \mathbb{R}^D$ by optimizing an image parameterization I to maximize the cosine similarity between \mathbf{a} and the activation \mathbf{a}' obtained from forward propagating I up to the specified layer. Each visualization is optimized independently.

In the Synthetic-RF variation we optimize a **receptive field size** image parameterization I_{RF} to approximately produce \mathbf{a} at a **single spatial position** in the layer’s feature map. Let $\mathbf{a}' \in \mathbb{R}^D$ be the activation obtained from forward propagating I_{RF} up to the layer. The visualization I_{RF}^* is as follows:

$$I_{\text{RF}}^* = \arg \max_{I_{\text{RF}}} \left(\frac{\mathbf{a} \cdot \mathbf{a}'}{\|\mathbf{a}\| \|\mathbf{a}'\|} \right) \quad (6.1)$$

In the Synthetic-Full variation we are optimizing a **full size** image parameterization I_{Full} to approximately produce \mathbf{a} at **every spatial position** in the layer’s feature map. Let $\mathbf{a}' \in \mathbb{R}^{H \times W \times D}$ be the activation obtained from forward propagating I_{Full} up to the layer. The visualization I_{Full}^* is as follows:

$$I_{\text{Full}}^* = \arg \max_{I_{\text{Full}}} \left(\frac{1}{HW} \sum_i^H \sum_j^W \frac{\mathbf{a} \cdot \mathbf{a}'_{ij}}{\|\mathbf{a}\| \|\mathbf{a}'_{ij}\|} \right) \quad (6.2)$$

The torch-lucent library was used for visualization.¹ Following Olah et al. [54], we improved image interpretability by applying small random affine transformations to I at each optimization step, and using the Fourier basis with all frequencies scaled to have the same energy as our image parameterization I . It is worth noting that the space of all images is vast and only a tiny fraction of image space corresponds to natural images. The resulting feature visualizations I^* are more synthetic and unnatural, containing many artifacts and potential distractors. Nonetheless, the feature visualizations sometimes contain interpretable image features that can be related back to the natural image space.

Natural visualization

To directly visualize activations in natural image space, we visualize a point in activation space $\mathbf{a} \in \mathbb{R}^D$ by displaying the k nearest receptive-field sized image patches that have the lowest ℓ_2 distance in activation space to the point \mathbf{a} when forward propagated. We perform this search over the N receptive field-sized image patches in the training set from which we sampled activations. It is possible that adjacent points along a PC share the same nearest neighbor (e.g. bottom row Figure 6.3).

For early vision layers where the size of the receptive field is small, these visualizations are quite interpretable, as can be seen in Figure 6.4. However, for deeper layers with larger receptive fields, it become more difficult to interpret what features the Natural-RF visualizations have in common because there is more area for distractors to be present. To help alleviate this problem, we highlight the regions of the Natural-RF visualizations that strongly affect the proximity of the representation to the point \mathbf{a} in representation space that we are visualizing (see Supplementary B.2).

Borowski *et al.* [7] conducted a user study and found natural image explanations to be more interpretable than optimized feature visualizations [54]. Using both synthetic and natural image visualizations incorporates the strengths of both approaches.

¹<https://github.com/greentfrapp/lucent>

6.2.4 Interpret visualizations

When visualizing activation space points we expect to see meaningful changes as we move along a large eigenvalue PC. For example, the angle of edges may change from horizontal to vertical, the color may change from orange to blue, patterns may change from vertical/horizontal to diagonal, etc. This can help give an idea of what features the DNN has learned to extract from the input to make classification decisions. We provide example visualizations to illustrate such changes (Figures 6.3 and 6.4 and Supplementary Material). To quantify the interpretability of each dimension, we report the results of a systematic study with human participants (see Section 6.3.3).

6.3 Results

In this work we study the PyTorch [57] implementation of AlexNet [39] pretrained on ImageNet [14]. AlexNet comprises five convolutional layers (conv1, conv2, conv3, conv4, and conv5) followed by three fully-connected layers (fc1, fc2, and fc3) (more details in Supplementary B.1).

6.3.1 Examples of principal component visualizations

Figure 6.3 shows an example of three visualization methods visualizing a PC. Each of the synthetic visualizations help provide additional context for interpreting the natural visualization. It is difficult to exhaustively show our visualizations in this format so we have chosen to show the Natural-RF visualizations of the top 6 PCs for each AlexNet layer in Supplementary B.9. All visualizations can be interactively viewed through our **anonymous** interactive demo ². The PCs of conv1 and conv2 capture low level features such as brightness, color, frequency, angle, phase, and color center surrounds.

The PCs for conv1 capture features such as brightness, Gabor filter phase, color contrasts, and color center surrounds (see Figure 6.4). The PCs for conv2 capture features such as texture frequency, line orientation, and color (see Supplementary Figure B.10). These results are consistent with the findings of a study on early convolutional layers [53]. The PCs for conv3, conv4, and conv5 seem to capture more high level features such as textures and objects. Finally, the PCs of fc1, fc2, and fc3 seem to separate classes.

²<https://david-s-hippocampus.github.io/deep-representations-pca>

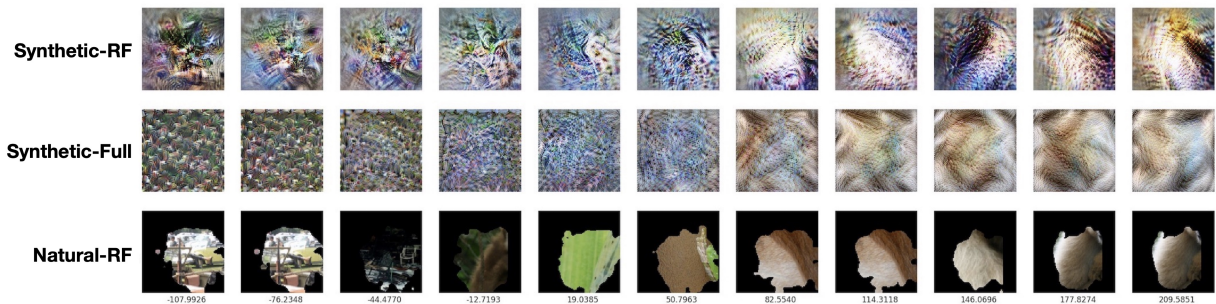


Figure 6.3: The full visualization for the fourth principal component of conv4. Positive values along this component correspond with the presence of a fur texture, while negative values seem to correspond with well-defined straight edges in the foreground and foliage in the background. In the far right columns, each of the 3 visualization types provides a different visualization of fur.

6.3.2 Measures of completeness

Explanation completeness is an abstract concept that could be measured in a variety of ways. We keep the explanation approach constant, and compare the completeness of the subspace to which the explanation applies. We use two complementary measures of subspace completeness below.

Explanation completeness: Activation covariance structure

One measure of completeness is the fraction of activation variance explained by an activation space basis. Equation 6.3 defines the fraction of explained variance as the amount of variance contained along a set of p basis vectors divided by the total variance contained along all P basis vectors.

$$\text{Fraction of explained variance} = \frac{\sum_i^N \sum_j^p [A_{ij} - \frac{\sum_k^N A_{kj}}{N}]^2}{\sum_i^N \sum_j^P [A_{ij} - \frac{\sum_k^N A_{kj}}{N}]^2} \quad (6.3)$$

Figure 6.5 shows a comparison between the explained variance of the PCA basis and the neuron basis for AlexNet’s activations. Much of the activation variance is concentrated in the most important PCs whereas variance is far less concentrated in the neuron basis. For example, to explain 80% of the activation variance for conv5, one could either study the first 33 PCs, or the 187 highest variance neurons. This trend remains clear for every

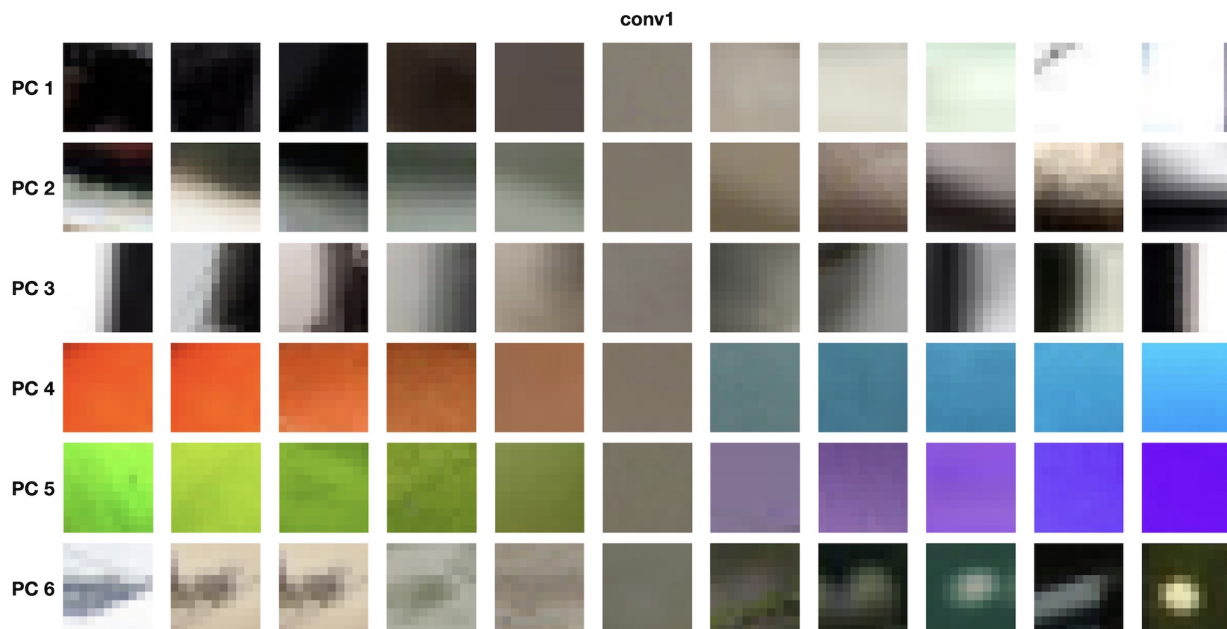


Figure 6.4: Natural-RF visualizations along the first 6 principal components of conv1. Brightness (PC 1), phase (PC 2,3), color contrast (PC 4,5), color-center-surround (PC 6).

layer. Clearly PCA is more efficient basis for activation space — dramatically fewer basis vectors are needed to explain a specified amount of activation variance. The success of the PCA is due to significant covariance between neurons in each layer. Without inter-neuron covariance, the PCA basis would be roughly as efficient as the standard neuron basis for explaining activation variance.

Explanation completeness: Activation ablation

Another quantitative measure of completeness is the amount of generalization performance that can be attributed to a set of basis vectors. To quantitatively test the importance of each basis, we ablate basis vectors (PCs or individual neurons) and observe how the validation accuracy of AlexNet pretrained on ImageNet degrades. We determine individual neuron importance by ablating each neuron individually, then assigning greater importance to neurons that caused larger decreases in validation accuracy (see Supplementary B.7). We ablate both neurons and PCs in order of importance and reverse order of importance, then measure the effect on AlexNet’s validation accuracy.

Figure 6.6 shows that for all layers after conv1, accuracy degrades faster when large

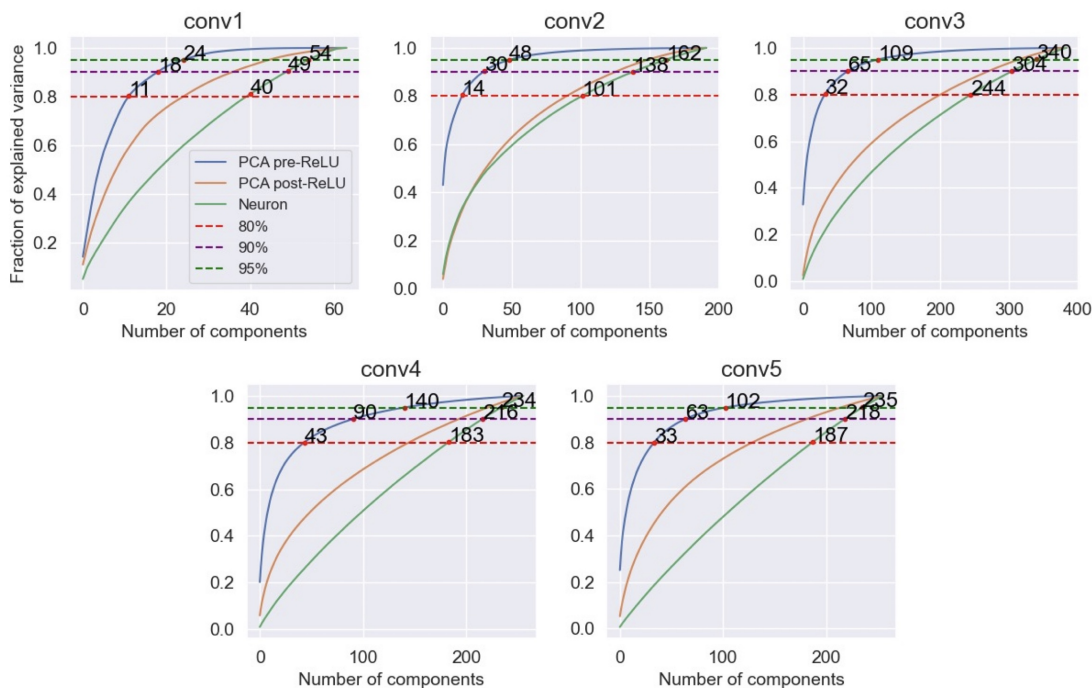


Figure 6.5: Cumulative sum of explained variance for each AlexNet convolutional layer. Both PCs and neurons are ordered by descending variance.

eigenvalue PCs are ablated compared to when the most important neurons are ablated. Conversely for all layers after conv2, when small eigenvalue PCs are ablated, they result in a smaller accuracy degradation than ablating the least important neurons. This is evidence that the most important PCs offer a more complete explanation of activation space than the most important neurons, and the least important PCs contribute the least towards explanation completeness. Zhou et al. [85] found that ablating random basis vectors from AlexNet conv5 had less of an effect on accuracy than ablating individual neurons. We show that ablating the most important PCs has a greater effect on accuracy than individual neurons, and thus also has a greater effect than ablating random directions for conv5.

6.3.3 Explanation Interpretability: Human Validation User Study

We conducted a user study to validate that humans can indeed interpret coherent stimuli across the visualized components. The core logic of our task is simple: If humans can detect

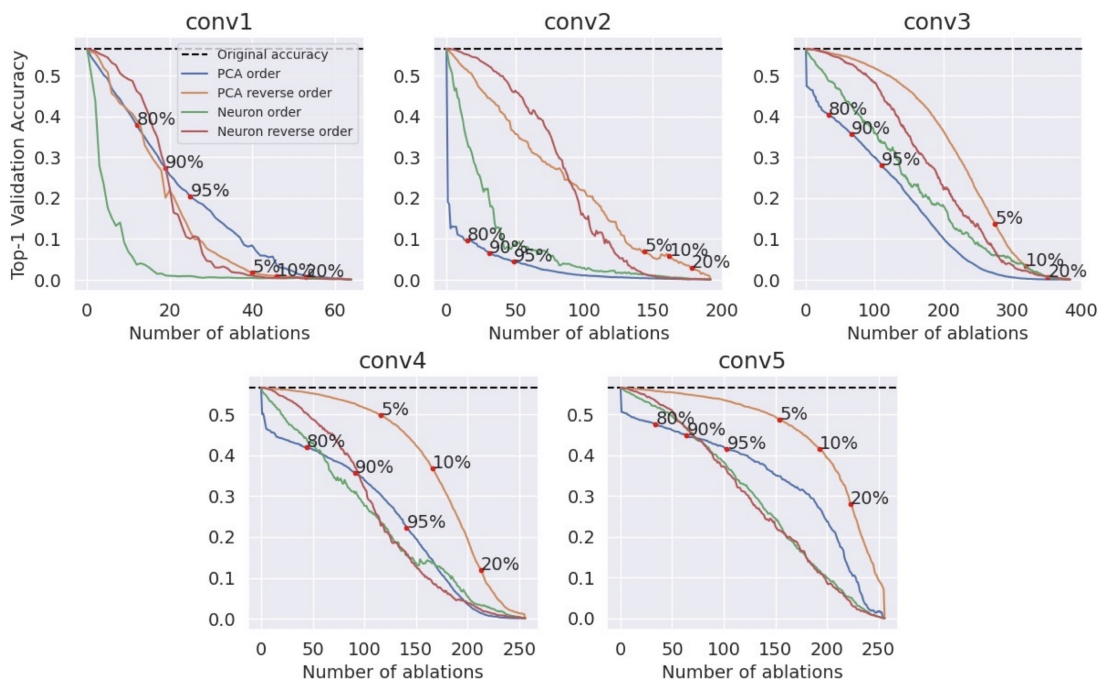


Figure 6.6: For each of AlexNet’s five convolutional layers, we ablate basis vectors in activation space and measure the effect on ImageNet top-1 validation accuracy. "PC order" (blue line) and "PC reverse order" (orange line) ablated PCs in descending and ascending order of eigenvalues respectively. "Neuron order" (green line) and "Neuron reverse order" (red line) ablated neurons in descending and ascending order of their individual effect on validation accuracy respectively. The points where PCs are responsible for specific percentages of explained variance are also annotated.

a coherent stimulus feature across the visualizations, they should be able to tell when they are looking at one of our visualizations versus a version of the same visualization with randomly-reordered points along the component. If observers cannot accurately determine which one is random, then they cannot really interpret the stimulus dimension. We tested the following hypotheses: H_1 - PCA visualizations are more interpretable than neuron-based visualizations; H_2 - Visualizations from shallow layers are more interpretable than deeper layers; H_3 - Interpretability is greater for more important bases, but only for PCA.

Method

We recruited 24 participants from Prolific in exchange for £6.6 GBP/hr. All participants provided informed consent to use their anonymized data in accordance with the institutional IRB. For additional details required to replicate the study, see Supplementary Materials. Participants were told they would see a display with two visualizations, and that one of them would be in a scrambled order. Their task would be to identify which of the two visualizations displays a coherent transition from left to right (see Figure B.3). We tested 144 components over AlexNet’s five convolutional layers. Components were sampled without replacement with probabilities equal to each PC’s explained variance ratio. We chose PCs this way because we expected interpretability to fall off quickly; having more visualizations from higher-order components should yield a more informative description of interpretability. For neuron-based visualizations, we selected the 144 most important neurons as determined by the ablation study, matching the number per layer as in the PCA visualizations. Note this is a conservative comparison of bases in favour of the neuron method. Stimuli were presented in random order. The position (top/bottom) of the original and scrambled stimuli was randomized. A visualization consisted of three rows of natural image snippets from the three nearest neighbours in activation space to points along each basis. The scrambled versions contained identical snippets in pseudo-random order (see Supplementary section B.6 for detail).

Results

Mean accuracy is depicted in Figure 6.7. Visual inspection reveals that for PCA, interpretability is clearly a function of layer depth and PC importance. Layer conv1 has all interpretable components, and from conv2 on there is a steep drop off with component that is more pronounced with layer depth. For neuron-based visualizations, there appears to be no relationship between neuron importance and interpretability and while conv1 appears interpretable, the rest drop off quickly. We support these claims with the following statistical analyses.

H_1 : In a paired-samples t -test matching basis rank order, PCA components were significantly more interpretable than neuron bases: $t(142) = 3.13$, $p = .002$. Note the neuron bases are from the 144 most important neurons determined via ablation, whereas the 144 PCA components were sampled without replacement with probabilities equal to each PC’s explained variance ratio, making this comparison unfair in favour of the neurons; this is very compelling evidence for the superiority of PCA interpretability. For the remaining hypotheses, we entered the data into a logistic generalized linear model with accuracy as

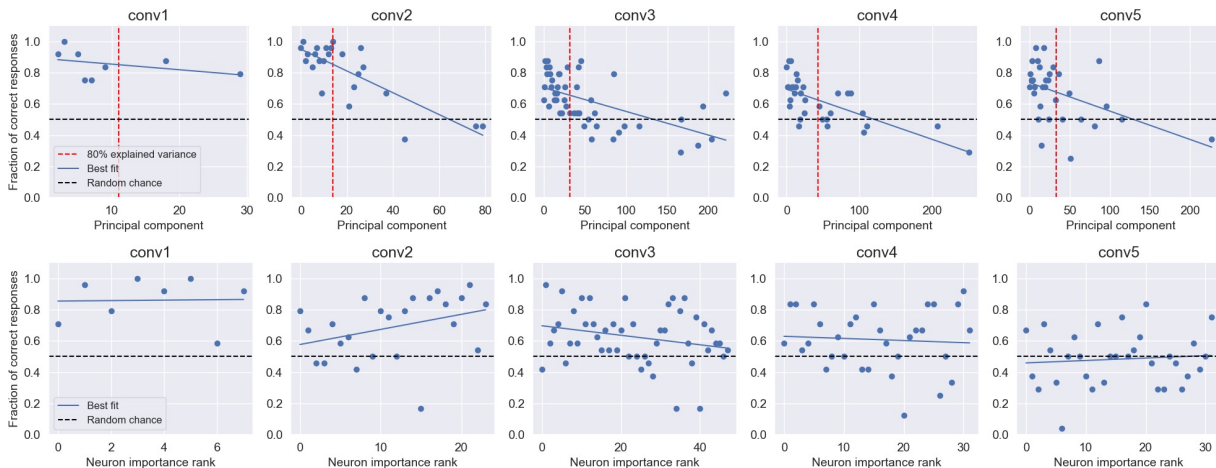


Figure 6.7: User study results for PCA (top) and neuron (bottom) basis visualizations.

the dependent variable and method (PCA vs neuron; categorical), layer (ordinal), and basis (ordinal) as predictors, including interactions. Note we rank-ordered neuron importance and PCA component, so these comparisons again give an unfair advantage to the neuron methods, as we took the most important neurons determined via ablation and pitted them against a non-linear sample of best components. Note also that we violate the independence assumption, however this should make this analysis more conservative. H_2 : There was a significant negative effect of layer in the regression ($B(1) = -.116, p < .001$), indicating deeper layers produced visualizations with worse interpretability. H_3 : There was a significant interaction between method and basis ($B(1) = .049, p < .001$), indicating that performance decreased with less important bases, and that this effect was more pronounced for PCA versus neuron.

6.4 Limitations

While we believe our work can provide practically useful explanations for DNNs, all explainability methods carry the risk that users will be overconfident in the explanations. To help mitigate this risk, we highlight three main limitations to our method of studying PCs of activation space. First, we do not consider how PCs interact in activation space because we study them individually. This is an important direction for future work because most stimuli are represented by multiple PCs. There are non-linear phenomenon in activation space but the PCA basis only provides a linear view of the underlying non-linear manifold

[20]. For example, tuning curves in biological neural networks are captured by non-linear manifolds [30, 52, 21, 10, 20]. Non-linear dimension reduction techniques may capture non-linear manifolds more accurately than linear techniques like PCA. If an unsupervised non-linear method could identify the neural manifolds of DNNs, it would be exciting to see if neural manifolds in biological neuron populations could also be identified. It is unclear what small eigenvalue PCs represent in activation space. Small eigenvalue PCs may be interpretable when considered in the context of their interaction with large eigenvalue PCs. They also may represent noise in activation space. Although they have uninterpretable visualizations and explain little variance, they still affect AlexNet’s validation accuracy when ablated. Thus, small eigenvalue PCs play some role in network function. They may be needed for representing long-tail inputs that appear infrequently in the training set. They could also be artifacts of the non-linear manifolds in activation space.

6.5 Conclusion

New insights into deep representations may be gained by visualizing PCs in activation space. PCA is a simple unsupervised method for identifying basis vectors for a layer’s activation space and analyzing high-dimensional distributed DNN representations. Large eigenvalue PCs correspond to semantically meaningful features in input space. Furthermore, most of the activation variance may be explained by dramatically fewer PCs than neurons. Large eigenvalue PCs are more interpretable and cause a large drop in validation accuracy when ablated. Similarly, small eigenvalue PCs are uninterpretable, and they have a small effect on validation accuracy when ablated. Our method has many advantages: it provides relatively interpretable and complete explanations of deep representations, is easy to apply to any CNN or image dataset, considers the distributed nature of deep representations, and is computationally inexpensive.

Chapter 7

Conclusions

This thesis shows that one can learn more about CNN representations by identifying basis vectors in activation space and visualizing points along the basis vectors to relate activation variance back to input variance. The first article evaluates the PCA, ICA, NMF, and LLE bases for activation space and shows that both PCA and ICA could identify tuning dimensions present in the early vision layers of InceptionV1 [73] as well as a synthetic Gabor bank. The second article built upon the first article by adding an improved visualization method that enabled the study of representations of deeper network layers with more confidence. The article demonstrated that visualizations along the principal components of AlexNet’s activations offer a more complete and interpretable explanation than visualizing individual neurons. This thesis contributes novel methods that can help researchers understand the representations that DNNs learn.

7.1 Future work

This work identifies several areas for future work that can further advance the field of XAI. The methods in the first and second articles are proposed as global explainability methods. The methods can be adapted to provide instance-specific counterfactual explanations. By forward propagating a data instance through a DNN up to a layer, the values along each basis vector for activation space can be obtained. To provide a counterfactual explanation one could adjust the value along a basis vector, then forward propagate the activations up to the DNN’s output layer to study how changing values along basis vectors affects the output.

The methods in this thesis could also be extended to study basis vectors of reinforcement learning, natural language processing, and graph learning models. In a reinforcement learning context, a DNN layer’s activation space can be sampled by saving the activations at each simulation timestep, aggregated over several task episodes. Sampling a layer’s activation space in natural language processing and graph learning contexts would be similar to a computer vision context: measure the activation produced by every data instance in the training set. Synthetic visualizations would require a method for visualizing an activation \mathbf{a} by optimizing an input parameterization to reproduce \mathbf{a} . The input parameterization would be a state vector, word vector, or graph for reinforcement learning, natural language processing, and graph learning respectively. Natural visualizations could be produced by displaying the data instances that, when forward propagated, produce an activation closest to the activation \mathbf{a} being visualized.

As mentioned in Section 6.4, when studying deep representations using the PCA basis, one is using a linear view to study the non-linear phenomenon present in the representation. Instead of PCA, if a non-linear unsupervised dimensionality reduction technique could be successfully applied to deep representations, the resulting non-linear basis vectors may align well with the non-linear phenomenon present in the representation. For example, the basis vectors may capture the non-linear manifolds that correspond to the exact tuning dimensions of a DNN layer. There are three main issues with applying non-linear dimensionality reduction methods to large sets of randomly sampled activations. The first issue is that many nonlinear methods have prohibitively high computation and memory requirements. This may be resolved by selecting more scalable methods, fitting to a smaller sample of activations, or gaining access to better hardware. The second issue is that many nonlinear dimensionality reduction methods do not possess an analytical inverse transform. This issue could be resolved by selecting methods with analytical inverse transforms such as kernel PCA (kPCA) [66] or general incompressible flow network (GIN) [71]. Some methods such as uniform manifold approximation and projection (UMAP) possess approximate inverse transforms. The third issue is that the dimensionality reduction problem is quite difficult and the data is quite noisy. In Section 4.4 locally linear embedding (LLE) [62] was fit to pre-trained InceptionV1 [73] activations and Gabor bank activations with no success. Unpublished preliminary experiments with nonlinear dimensionality reduction techniques such as UMAP [48], t-distributed stochastic neighbor embedding (t-SNE) [76], β variational autoencoder (β -VAE) [28], and GIN [71] were performed with no success unfortunately. Nonlinear dimensionality reduction methods are notoriously sensitive to hyperparameters and noise so it is possible that more rigorous hyperparameter tuning and careful sampling of activations to reduce noise could lead to success. A successful non-linear unsupervised dimensionality reduction technique could provide a more complete basis and lead to more

interpretable visualizations.

The interpretability of the explanations in this thesis could be increased by using more sophisticated visualization techniques. Visualization techniques that involve generative models [51, 61] offer the highest visualization quality at the cost of needing to first train generative models to invert the activations of some layer. The additional computational cost of training a generative model could be prohibitively expensive for many users but it would result in more interpretable explanations.

References

- [1] Jay Alammari. Interfaces for explaining transformer language models, 2020.
- [2] Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. Evaluating saliency map explanations for convolutional neural networks: A user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces, IUI '20*, page 275–285, New York, NY, USA, 2020. Association for Computing Machinery.
- [3] Mathieu Aubry and Bryan C. Russell. Understanding deep features with computer-generated imagery. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2875–2883, 2015.
- [4] David GT Barrett, Ari S Morcos, and Jakob H Macke. Analyzing biological and artificial neural networks: challenges with opportunities for synergy? *Current Opinion in Neurobiology*, 55:55–64, 2019. Machine Learning, Big Data, and Neuroscience.
- [5] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*, 2017.
- [6] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 2020.
- [7] Judy Borowski, Roland Simon Zimmermann, Judith Schepers, Robert Geirhos, Thomas S. A. Wallis, Matthias Bethge, and Wieland Brendel. Natural images are more informative for interpreting {cnn} activations than synthetic feature visualizations. In *NeurIPS 2020 Workshop SVRHM*, 2020.

- [8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.
- [9] Santiago A. Cadena, George H. Denfield, Edgar Y. Walker, Leon A. Gatys, Andreas S. Tolias, Matthias Bethge, and Alexander S. Ecker. Deep convolutional models improve predictions of macaque V1 responses to natural images. *PLoS Computational Biology*, 15(4):e1006897, 04 2019.
- [10] Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 2020. <https://distill.pub/2020/circuits/curve-detectors>.
- [11] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 2019. <https://distill.pub/2019/activation-atlas>.
- [12] Andrzej Cichocki and Anh-Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions*, 92-A:708–721, 03 2009.
- [13] John P. Cunningham and Byron M. Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, Nov 2014.
- [14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [16] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017.
- [17] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations, 2019.
- [18] Patrick Esser, Robin Rombach, and Björn Ommer. A disentangling invertible interpretation network for explaining latent representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

- [19] Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8730–8738, 2018.
- [20] Juan A. Gallego, Matthew G. Perich, Lee E. Miller, and Sara A. Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, 2017.
- [21] Apostolos P. Georgopoulos, Andrew B. Schwartz, and Ronald E. Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.
- [22] Amirata Ghorbani, James Wexler, James Zou, and Been Kim. Towards automatic concept-based explanations, 2019.
- [23] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89, 2018.
- [24] David Gunning and David Aha. Darpa’s explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, Jun. 2019.
- [25] Peter Hase and Mohit Bansal. Evaluating explainable ai: Which algorithmic explanations help users predict model behavior?, 2020.
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [27] Zhenliang He, Meina Kan, and Shiguang Shan. Eigengan: Layer-wise eigen-learning for gans. *arXiv:2104.12476*, 2021.
- [28] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.
- [29] Geoffrey E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1):185–234, 1989.
- [30] David H. Hubel and Torsten N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *The Journal of Physiology*, 160(1):106–154, 1962.

- [31] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411 – 430, 2000.
- [32] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Proc. NeurIPS*, 2020.
- [33] Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks. In *International Conference on Learning Representations*, 2020.
- [34] Andrej Karpathy. t-sne visualization of cnn codes. <https://cs.stanford.edu/people/karpathy/cnembed/>, 2014.
- [35] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020.
- [36] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677. PMLR, 10–15 Jul 2018.
- [37] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un)reliability of saliency methods, 2017.
- [38] Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. Representational similarity analysis – connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4, 02 2008.
- [39] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *ArXiv*, abs/1404.5997, 2014.
- [40] Oran Lang, Yossi Gandelsman, Michal Yarom, Yoav Wald, Gal Elidan, Avinatan Hassidim, William T. Freeman, Phillip Isola, Amir Globerson, Michal Irani, and Inbar Mosseri. Explaining in style: Training a gan to explain a classifier in stylespace, 2021.
- [41] Matthew L. Leavitt and Ari Morcos. Towards falsifiable interpretability research, 2020.
- [42] Matthew L Leavitt and Ari S. Morcos. Selectivity considered harmful: evaluating the causal impact of class selectivity in {dnn}s. In *International Conference on Learning Representations*, 2021.

- [43] Yann Lecun. *Generalization and network design strategies*. Elsevier, 1989.
- [44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.
- [45] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. *International Journal of Computer Vision*, 127(5):456–476, May 2019.
- [46] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.
- [47] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196, 2015.
- [48] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.
- [49] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17153–17163. Curran Associates, Inc., 2020.
- [50] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *Visualization for Deep Learning workshop, International Conference in Machine Learning*, 2016. arXiv preprint arXiv:1602.03616.
- [51] Anh Mai Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *CoRR*, abs/1605.09304, 2016.
- [52] Harris Nover, Charles H. Anderson, and Gregory C. DeAngelis. A logarithmic, scale-invariant representation of speed in macaque middle temporal area accounts for speed discrimination performance. *Journal of Neuroscience*, 25(43):10049–10060, 2005.
- [53] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. An overview of early vision in inceptionv1. *Distill*, 2020. <https://distill.pub/2020/circuits/early-vision>.

- [54] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. <https://distill.pub/2017/feature-visualization>.
- [55] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. <https://distill.pub/2018/building-blocks>.
- [56] Bruno Olshausen and David Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–9, 07 1996.
- [57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.
- [58] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [59] Carlos Ponce, Will Xiao, Peter Schade, Till Hartmann, Gabriel Kreiman, and Margaret Livingstone. Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences. *Cell*, 177:999–1009.e10, 05 2019.
- [60] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.
- [61] Robin Rombach, Patrick Esser, and Björn Ommer. Making sense of cnns: Interpreting deep representations & their invariances with inns. In *Proceedings of the European Conference on Computer Vision*, 2020.
- [62] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
- [63] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986.

- [64] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [65] Abigail A. Russo, Ramin Khajeh, Sean R. Bittner, Sean M. Perkins, John P. Cunningham, L.F. Abbott, and Mark M. Churchland. Neural trajectories in the supplementary motor area and motor cortex exhibit distinct geometries, compatible with different classes of computation. *Neuron*, 107(4):745–758.e6, 2020.
- [66] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. *Kernel Principal Component Analysis*, page 327–352. MIT Press, Cambridge, MA, USA, 1999.
- [67] Hua Shen and Ting-Hao Kenneth Huang. How useful are the machine-generated interpretations to general users? a human evaluation on guessing the incorrectly predicted labels, 2020.
- [68] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *CVPR*, 2021.
- [69] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.
- [70] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.
- [71] Peter Sorrenson, Carsten Rother, and Ullrich Köthe. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). In *International Conference on Learning Representations*, 2020.
- [72] Mark Stopfer, Vivek Jayaraman, and Gilles Laurent. Intensity versus identity coding in an olfactory system. *Neuron*, 39(6):991–1004, Sep 2003.
- [73] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

- [74] Keiji Tanaka. Inferotemporal cortex and object vision. *Annual Review of Neuroscience*, 19(1):109–139, 1996. PMID: 8833438.
- [75] Bryan Tripp. Approximating the Architecture of Visual Cortex in a Convolutional Network. *Neural Computation*, 31(8):1551–1591, 08 2019.
- [76] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.
- [77] Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space, 2020.
- [78] Jianyu Wang, Zhishuai Zhang, Cihang Xie, Vittal Premachandran, and Alan Yuille. Unsupervised learning of object semantic parts from internal states of cnns by population encoding, 2016.
- [79] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 2020.
- [80] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20554–20565. Curran Associates, Inc., 2020.
- [81] Byron M. Yu, John P. Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V. Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of Neurophysiology*, 102(1):614–635, 2009. PMID: 19357332.
- [82] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.
- [83] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Proceedings of the 2011 International Conference on Computer Vision, ICCV '11*, page 2018–2025, USA, 2011. IEEE Computer Society.

- [84] Yimeng Zhang, Tai Sing Lee, Ming Li, Fang Liu, and Shiming Tang. Convolutional neural network models of V1 responses to complex patterns. *Journal of Computational Neuroscience*, 46(1):33–54, 02 2019.
- [85] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the importance of individual units in cnns via ablation, 2018.

APPENDICES

Appendix A

First article appendix

A.1 Comparing sampling methods

When exploring visualization methods there was no metric for interpretability so we qualitatively evaluated the visualizations based on their perceived interpretability. We found that sampling and visualizing 32 points along each tuning dimension was sufficient for observing the differences in the visualizations along the dimension.

We found that uniformly sampling m points between the observed minimum and maximum values along each tuning dimension produced the most informative visualizations. We also tried sampling so that the same proportion of observed points in activation space lie between each of the points along the tuning dimension. Since most activations are zero or near zero, this sampling strategy resulted in over-sampling the region near zero. Visualizations produced using these sampling methods are shown in Figure [A.1](#).

A.2 Gabor bank construction

A Gabor filter g is constructed by multiplying a sinusoid with a Gaussian as shown in Equation [A.1](#), where $x' = x \cos \theta + y \sin \theta$ and $y' = -x \sin \theta + y \cos \theta$.

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right) \quad (\text{A.1})$$

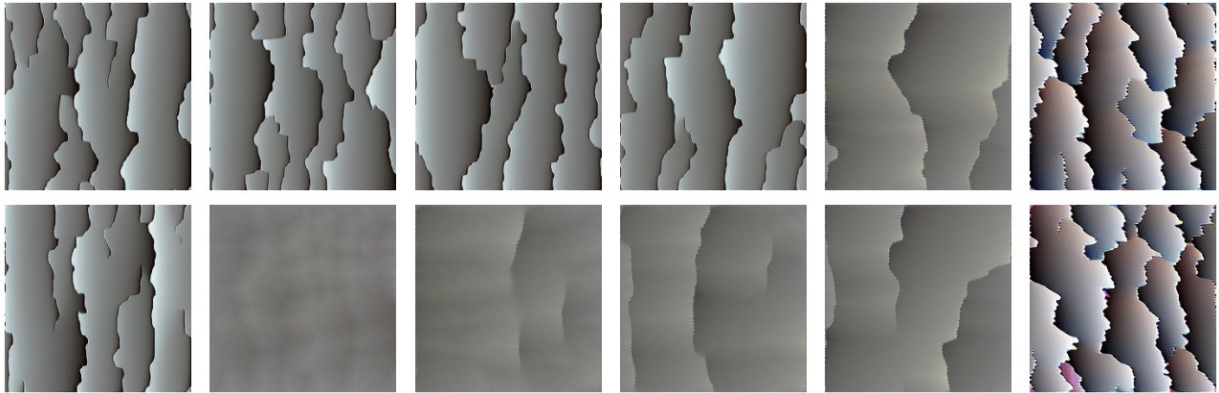


Figure A.1: Visualizations of the first ICA component fit to activations from the first layer of InceptionV1 (conv2d0). Top: Points sampled uniformly between the observed minimum and maximum values along the component. Bottom: Points sampled such that the same proportion of observed points in activation space lie between each of the points along the component. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

The Gabor bank was constructed by uniformly sampling (with a fixed random seed) frequency $(1/\lambda) \in [0.2, 0.3]$, angle $\theta \in [0, \frac{\pi}{2}]$, and phase $\psi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. We set $\gamma = 1$ and $\sigma = 2$. The Gabor filter weights and preferred stimuli are shown in Figure A.2.

A.3 Gabor bank tuning dimension visualizations

As discussed in Section 4.3.4, the visualizations must be interpreted by a human. The interpretation can be subjective so for transparency, we have included each of the visualizations that we interpreted as tuning dimensions. The results can also be explored through our interactive interface. The tuning dimension visualizations for PCA-3 and PCA-6 (Figure A.3), ICA-3 (Figure A.4), ICA-6 (Figure A.5), NMF-3 (Figure A.6), and LLE-3 (Figure A.7) are included.

A.4 InceptionV1 tuning dimension visualizations

As discussed in Section 4.3.4, the visualizations must be interpreted by a human. The interpretation can be subjective so for transparency, we have included each of the visualizations

that we interpreted as tuning dimensions. The results can also be explored through our interactive interface. The tuning dimension visualizations for `conv2d0` PCA-16 (Figure [A.8](#)), `conv2d0` ICA-16 (Figure [A.9](#)), `conv2d1` PCA-16 (Figure [A.10](#)), and `conv2d1` ICA-16 (Figure [A.11](#)) are included.

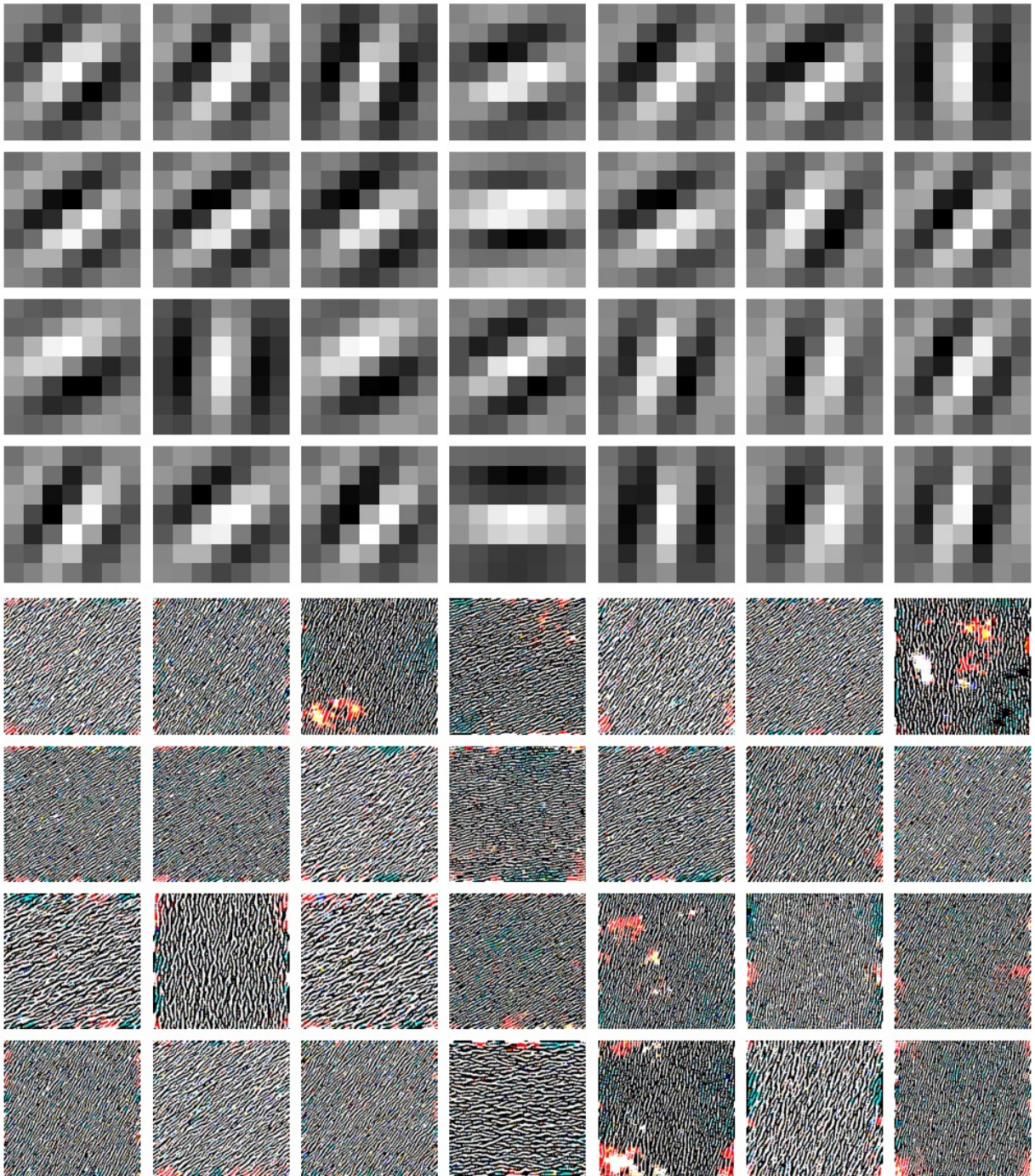


Figure A.2: Top: Gabor bank filter weights. The values in each of the RGB channels are equal. Bottom: Preferred stimuli of each Gabor filter found through feature visualization [54]. Best viewed in color.

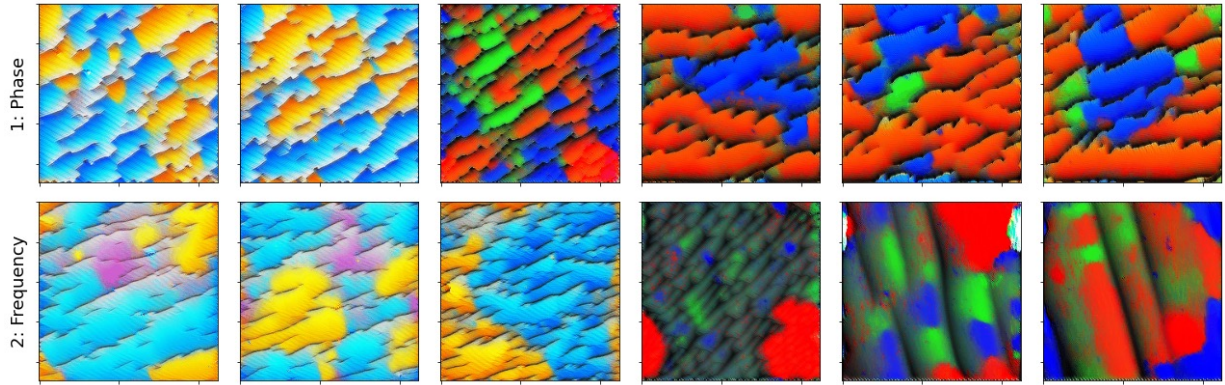


Figure A.3: Gabor bank PCA-3 and PCA-6 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Due to the ordering of PCA, the first 3 principal components are shared between PCA-3 and PCA-6. Best viewed in color.

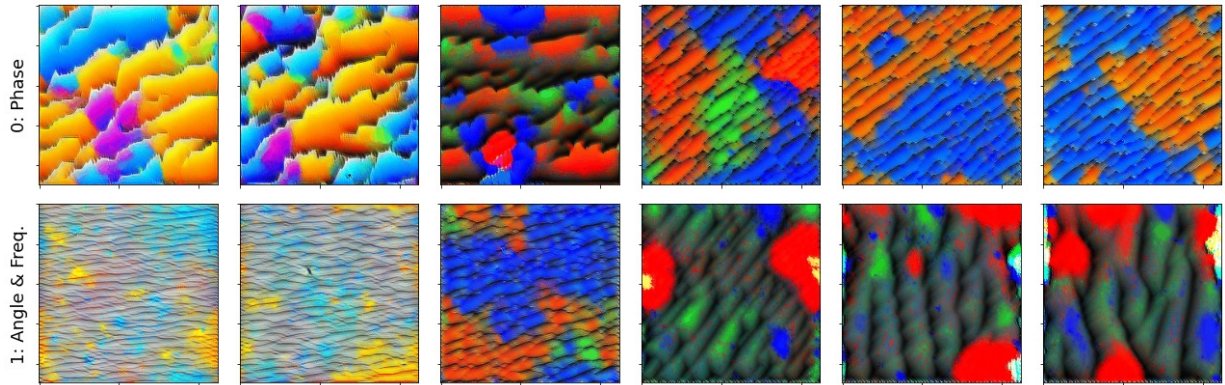


Figure A.4: Gabor bank ICA-3 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

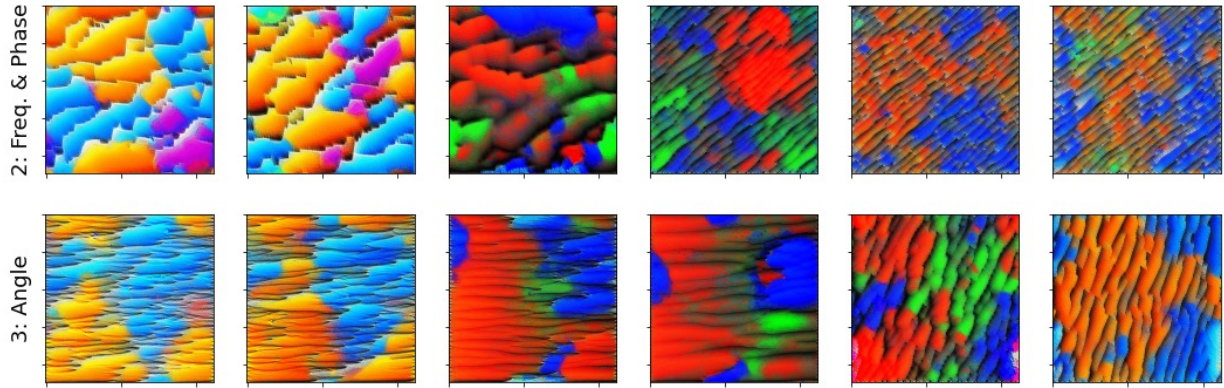


Figure A.5: Gabor bank ICA-6 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

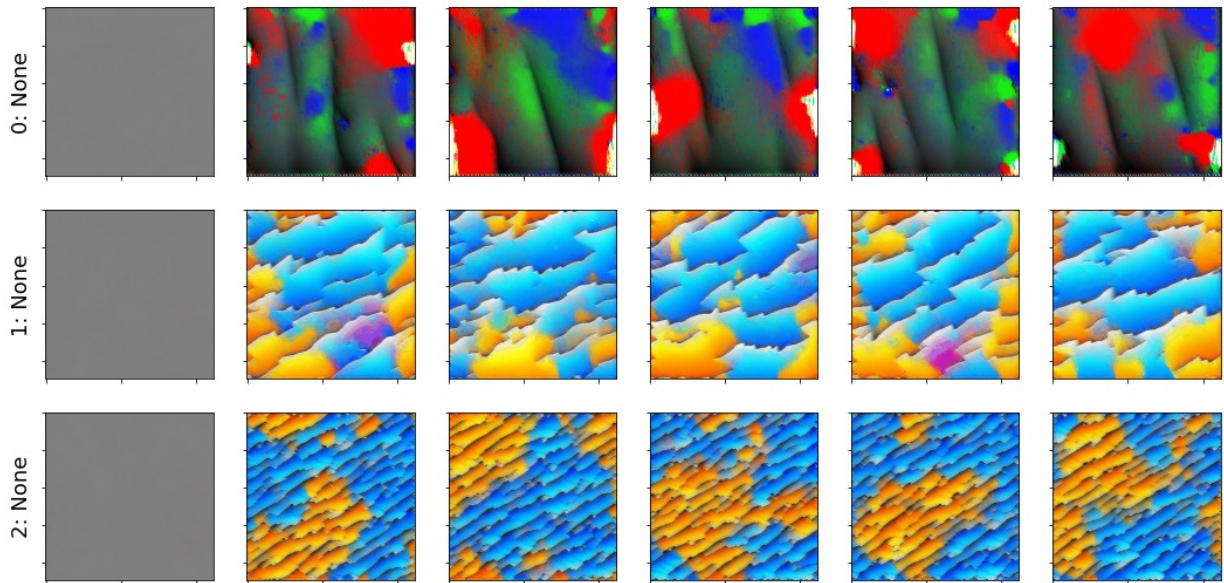


Figure A.6: Gabor bank NMF-3 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. No meaningful variation was observed along each tuning dimension so no tuning dimensions were interpreted. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

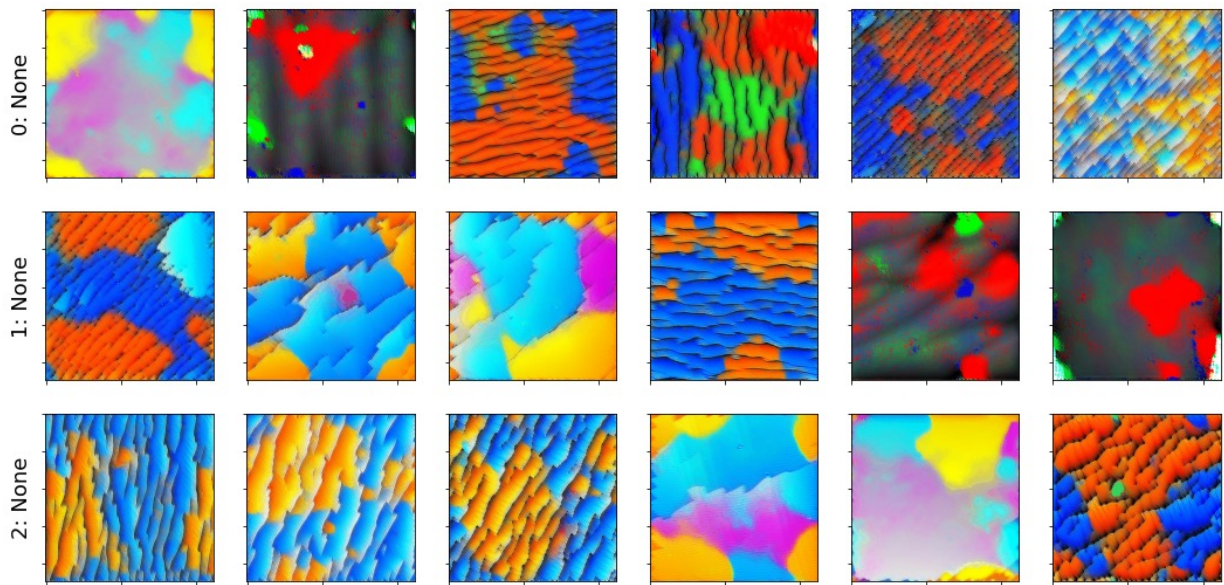


Figure A.7: Gabor bank LLE-3 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. No apparent pattern was observed along each tuning dimension so no tuning dimensions were interpreted. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

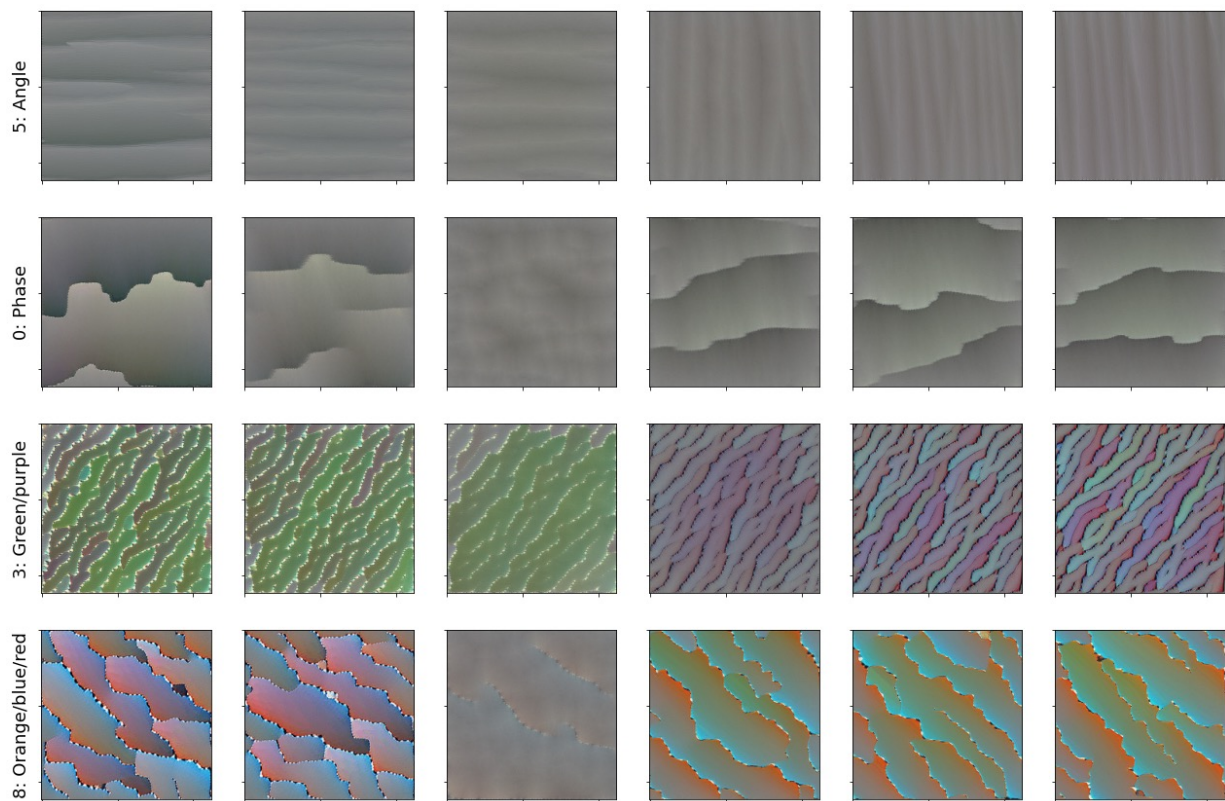


Figure A.8: conv2d0 PCA-16 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

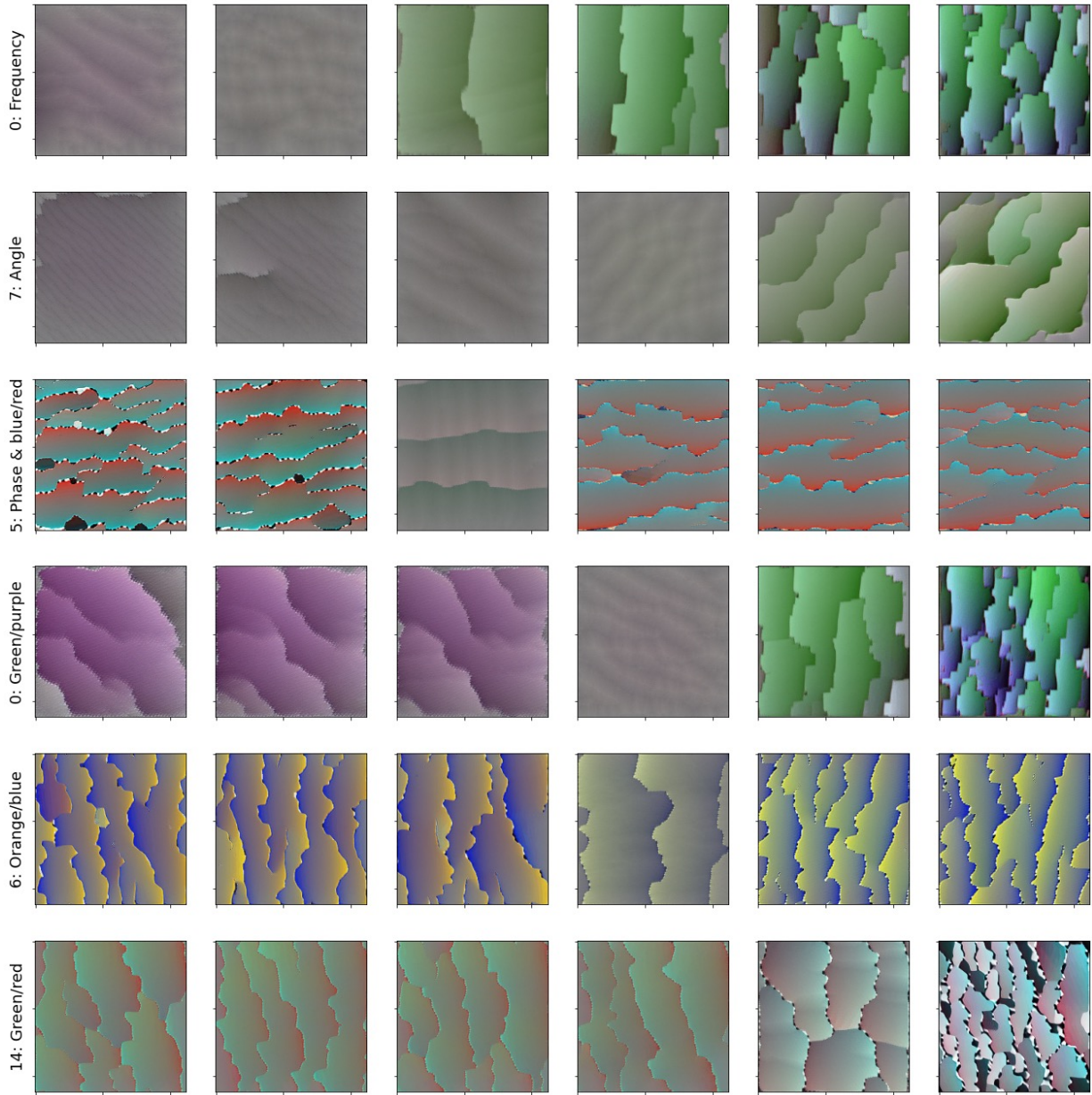


Figure A.9: conv2d0 ICA-16 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

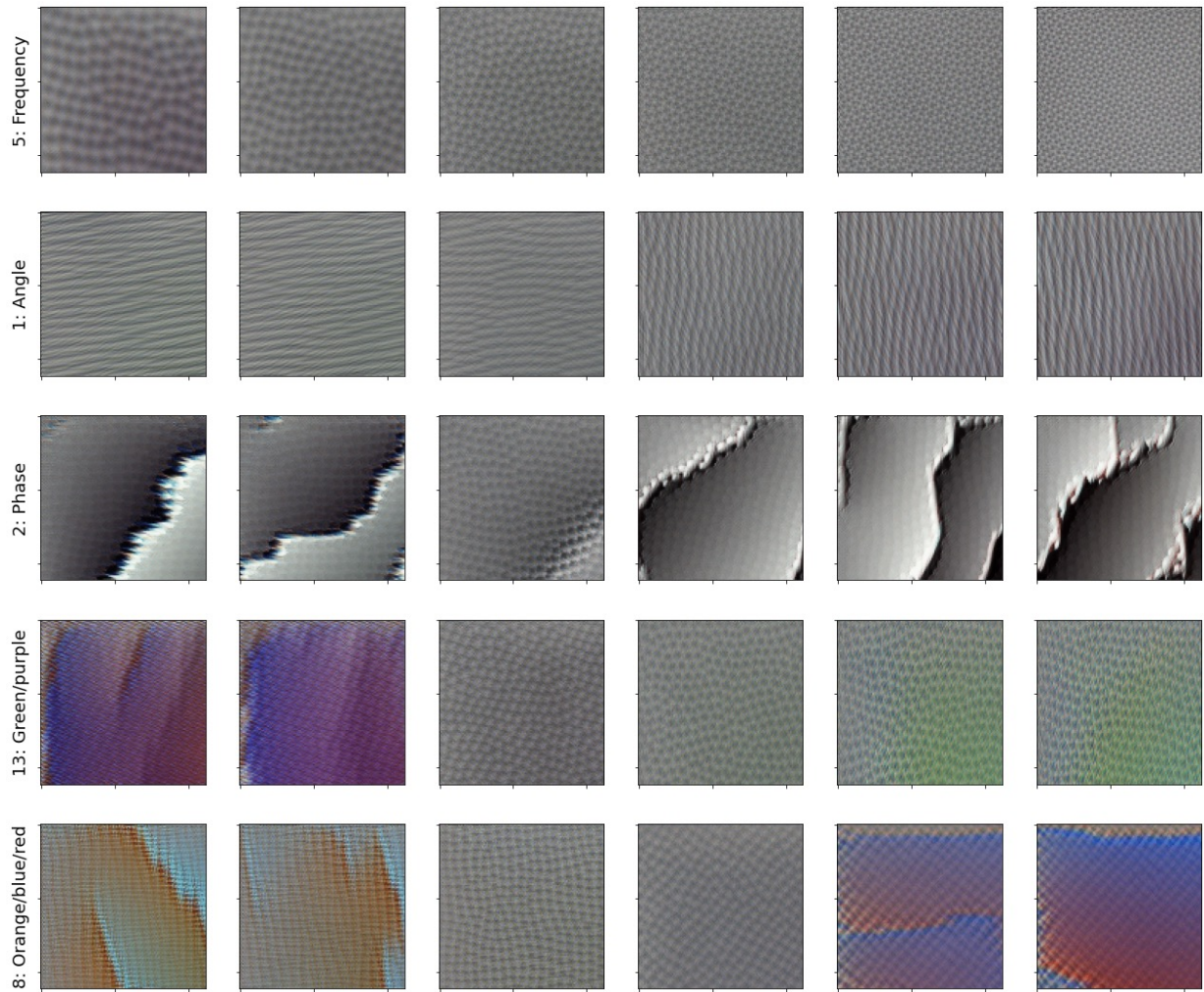


Figure A.10: conv2d1 PCA-16 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

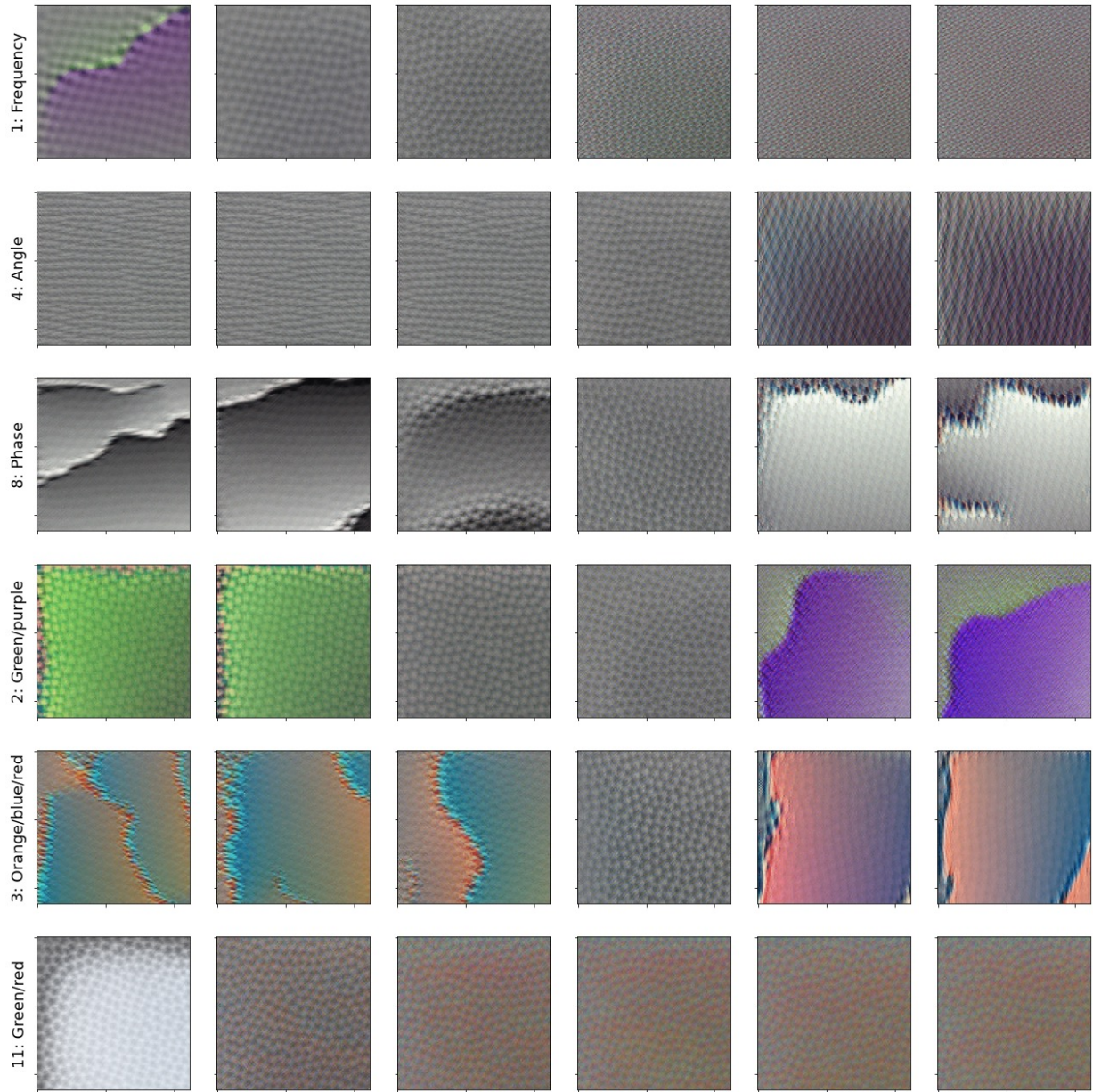


Figure A.11: conv2d1 ICA-16 tuning dimension visualizations with their component number and interpreted tuning dimension shown on the left. 32 points were visualized for each tuning dimension but only 6 are shown to improve readability. Best viewed in color.

Appendix B

Second article appendix

B.1 AlexNet architecture

The architecture for the PyTorch [57] implementation of AlexNet [39] that we used in this paper is shown in Table B.1.

B.2 Masking natural image regions that contribute strongly to activation similarity

As discussed in Section 6.2.3, the Natural-RF visualizations for deeper layers with larger receptive fields are more difficult to interpret than early layers with small receptive fields because there is more area in the natural image for distractors to be present. To help alleviate this problem, we highlight the regions of the Natural-RF visualizations that strongly affect the proximity of the representation to the point \mathbf{a} in representation space that we are visualizing.

Let H_I , W_I , and D_I be the height, width, and number of channels in each image patch I . The sensitivity gradient $G \in \mathbb{R}^{H_I \times W_I \times D_I}$ is the gradient of the ℓ_2 distance between the target point \mathbf{a} and \mathbf{a}' , the activation from forward propagating I , with respect to the pixels in the image patch I .

$$G = \frac{\partial}{\partial I} \|\mathbf{a} - \mathbf{a}'\|_2^2 \quad (\text{B.1})$$

Table B.1: Architecture for the PyTorch [57] implementation of AlexNet [39]. In the parameters column, k, s, p, and d are kernel size, stride, padding, and dropout probability respectively.

Layer name	Layer type	Activation	Parameters	Output shape
	Input			$3 \times 224 \times 224$
conv1	Conv2d	ReLU	k=11,s=4,p=2	$64 \times 55 \times 55$
	MaxPool2d		k=3,s=2,p=0	$64 \times 27 \times 27$
conv2	Conv2d	ReLU	k=5,s=1,p=2	$192 \times 27 \times 27$
	MaxPool2d		k=3,s=2,p=0	$192 \times 13 \times 13$
conv3	Conv2d	ReLU	k=3,s=1,p=1	$384 \times 13 \times 13$
conv4	Conv2d	ReLU	k=3,s=1,p=1	$256 \times 13 \times 13$
conv5	Conv2d	ReLU	k=3,s=1,p=1	$256 \times 13 \times 13$
	MaxPool2d		k=3,s=2,p=0	$256 \times 6 \times 6$
	AdaptiveAvgPool2d			$256 \times 6 \times 6$
	Flatten			9216
	Dropout		d=0.5	9216
fc1	Linear	ReLU		4096
	Dropout		d=0.5	4096
fc2	Linear	ReLU		4096
	Dropout		d=0.5	4096
fc3	Linear	ReLU		1000

Next we take the sum of the absolute gradient values over channels and apply a 2D Gaussian filter $g(x, y, \sigma)$, where $\sigma = 3$, to obtain $G_{\text{smooth}} \in \mathbb{R}^{H_I \times W_I}$ as follows:

$$G_{\text{smooth}} = g(x, y, \sigma) * \left(\left| \sum_d^{D_I} (G_d) \right| \right) \quad (\text{B.2})$$

To determine the Boolean image patch mask $M \in \mathbb{R}^{H_I \times W_I}$, we threshold G_{smooth} to be greater than one standard deviation of G_{smooth} as follows:

$$M = G_{\text{smooth}} > \text{std}(G_{\text{smooth}}) \quad (\text{B.3})$$

The resulting mask M is applied to the image patch I to hide regions of the image that do not strongly affect the distance $\|\mathbf{a} - \mathbf{a}'\|_2^2$.

B.3 Comparison of methods for sampling points along basis vectors

As discussed in Section 6.2.3, we sample and visualize m points along a basis vector with the goal of understanding how activation variance along the basis vector relates to input variance. We compared two strategies: sampling between the first and ninety-ninth percentile, or sampling between the minimum and maximum values along the basis vector. After studying several basis vector visualizations from each AlexNet layer, we concluded that sampling between the minimum and maximum values produced more interpretable visualizations.

In Figure B.1 we show a comparison of the two strategies for the second PC of conv1, which represents the Gabor phase of a vertical edge. Although the first and ninety-ninth percentile visualizations demonstrate the concept of Gabor phase, the extreme values along each basis vector produce visualizations that demonstrate the concept more distinctly.

B.4 Computational resources

Our computations were performed on machines with an 8 core Intel Xeon CPU, NVIDIA T4 GPU, and 64 GB of RAM. Sampling activations from an AlexNet layer for the ImageNet training set took approximately 2 hours. The runtime of fitting PCA to the sampled activations A is dependent on the dimensions of A , and can take anywhere from 11 seconds

for conv1 (64-dimensional) to 22 minutes for fc1 (4096-dimensional). Producing synthetic visualizations of 32 points along a basis vector takes approximately 40 seconds. Producing natural visualizations of 32 points along a basis vector, with 5 neighbors, takes approximately 4 seconds.

B.5 Distribution of principal component weights

Each PC is defined by a linear combination of neurons. The histograms of the PC weights for each AlexNet layer are shown in Figure B.2. The PC weights seem to contain more extreme magnitudes for conv1, suggesting that the representation of conv1 is less distributed than the deeper layers. This is consistent with the ablation study in Section 6.3.2 that showed validation accuracy decreases fastest when ablating neurons in conv1 instead of PCs. From conv1 to conv5, the representation becomes progressively more distributed (less extreme weight values). This also aligns with the ablation study in Section 6.3.2 that showed validation accuracy decreases fastest when ablating PCs instead of PCs in conv2 - conv5. Following conv5, fc1's representation becomes less distributed. Finally from fc1 to fc3, the representation once again becomes progressively more distributed.

B.6 Additional User Study Details

Full visual instructions are depicted in Figure B.3. Participants read these instructions at their own pace and proceeded with a button press.

The scrambled re-ordering for comparison images followed some constraints. When piloting the task with fully random scrambling, we discovered it was trivially easy if the NN row contained duplicate snippets, because they would not appear next to each other in the scrambled version (detecting two or more identical snippets not adjacent was a giveaway that it was the scrambled stimulus, and could be used as a signal to complete the task). To remedy this, comparison images were pseudorandomly ordered such that NN snippets that appeared multiple times within a row were required to be horizontally adjacent.

The full text instructions presented to each participant in the user study from Section 6.3.3 are shown in Figure B.3.

B.7 Neuron ablation

As a proxy for neuron importance, we ablated neurons **individually** and measured the effect on AlexNet’s ImageNet top-1 validation accuracy. Neurons that caused a larger reduction in validation accuracy were considered more important. The results of ablating each neuron from AlexNet’s five convolutional layers are shown in Figure B.4.

B.8 Neuron visualizations

To illustrate the interpretability of the neuron basis, we have included the Natural-RF visualizations of the first 6 neurons of each AlexNet convolutional layer in Figures B.5, B.6, B.7, B.8, and B.9. After conv1, the neuron basis becomes quite uninterpretable, as confirmed through our user study.

B.9 Additional principal component visualizations

It is difficult to exhaustively show our visualizations in this format so we have chosen to show the Natural-RF visualizations of the top 6 PCs for each AlexNet layer in Figures B.4, B.10, B.11, B.12, B.13, B.14, B.15, and B.16. Additionally, all visualizations can be interactively viewed through our **anonymous** interactive demo ¹.

¹<https://david-s-hippocampus.github.io/deep-representations-pca>

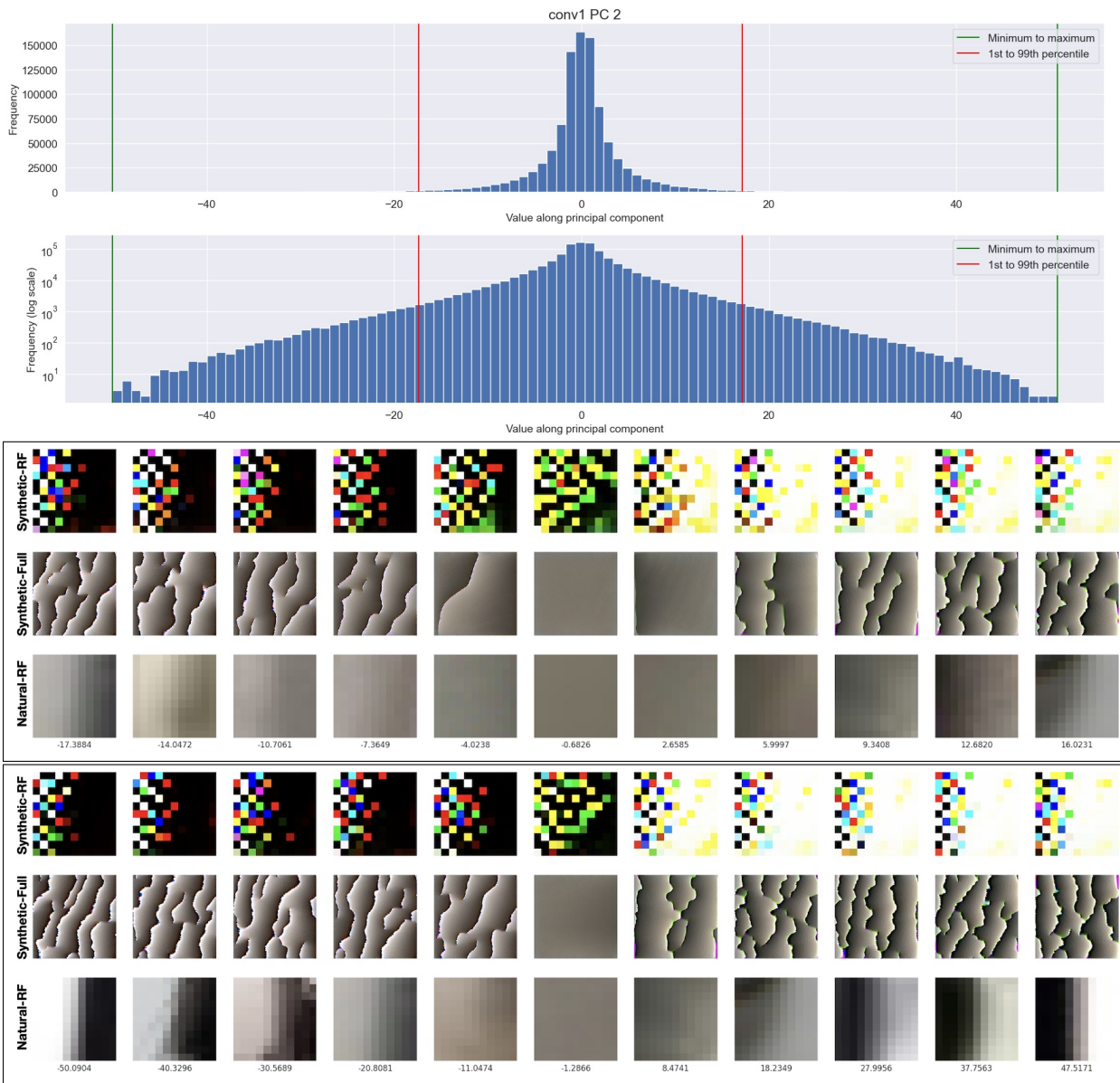


Figure B.1: **Top:** Histogram of activations from the ImageNet training set, projected onto the second PC (PC 2) of conv1. The interval defined by the minimum and maximum is shown in green. The interval defined by the first and ninety-ninth percentile is shown in red. **Middle and bottom:** Visualizations of points along PC 2 of conv1, uniformly sampled between the first and ninety-ninth percentile (middle), and uniformly sampled between the minimum and maximum (bottom).

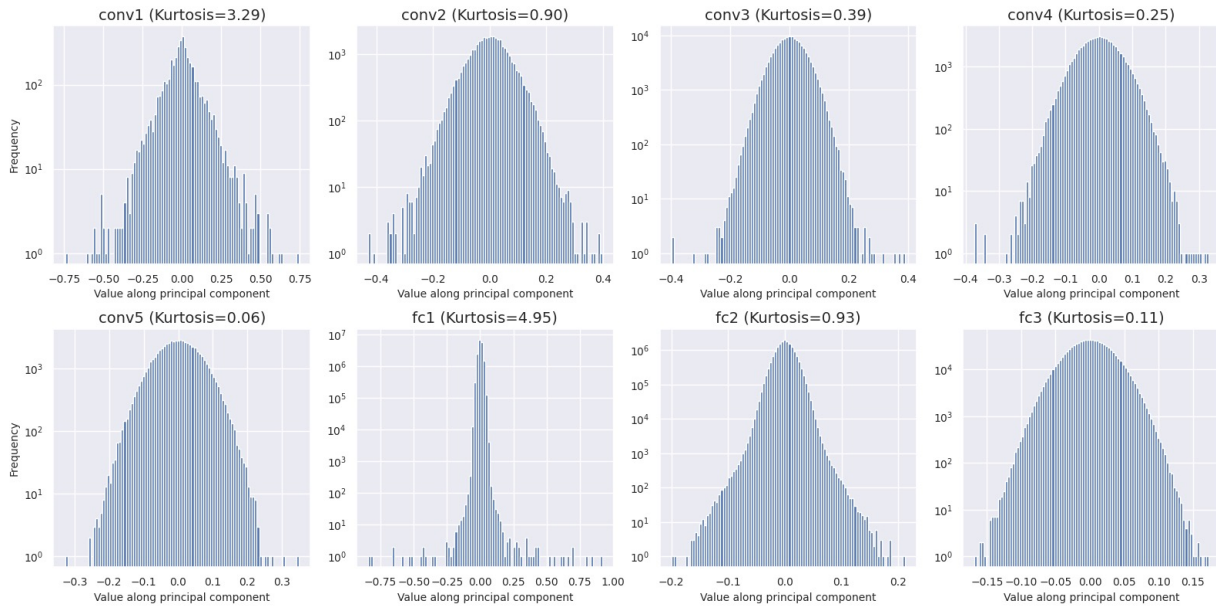



Figure B.2: Histograms of principal component weights for each AlexNet layer. The kurtosis for each distribution is also included in each plot title. A kurtosis of 0 corresponds to a normal distribution. From conv1 to conv5, the distribution of PC weights becomes progressively more similar to a normal distribution. Then, fc1 has a high kurtosis but from fc1 to fc3, the distribution of PC weights once again becomes progressively more similar to a normal distribution.

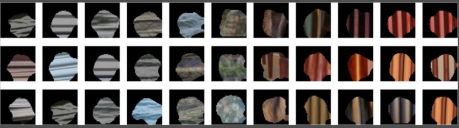
AI are designed to perform difficult tasks at a high level. Often, an AI will perform as good or better than a human on a certain task, but we, the users, lack an understanding of how the AI made a certain decision or behaved a certain way. In response, AI engineers are developing tools to explain how AI behaves.

In this experiment, we are studying AI for computer vision – that is, AI that sees and can identify objects in a scene. We have recently developed a tool that visualizes the features of a scene that the AI responds to – in other words, the input features that it “looks for” and that guide its object identification. This tool will show you a bunch of images that together span the range of some feature. For example, the images below show that one layer from our AI responds to lines of different orientation. The images on the left show horizontal lines transitioning to vertical on the right:



Your task in this study is to identify similar patterns. We want to know whether unbiased observers (you) can notice these patterns. On every trial, we will show you two versions of the same visualization: one that is the output from our visualization tool, and one that is the same series of images, but in scrambled order. Your task is to choose the original. In short, your job is to notice something in these images transitioning from left to right.

We’ve combined three different visualization techniques that are supposed to illustrate the same feature. Use them together to guide your decision. Together, they look like this:




In all three, you can make out horizontal contours on the left that transition to vertical ones on the right.


Here is another example. This one is easy! This visualization shows that some part of the AI is sensitive to red/orange inputs while others look for blue.



Remember, we are going to show you two versions of the same visualization: the original and a randomized version of the original. You need to choose the original one, so look for visualizations to show some continuity from left to right. Below, see what a trial might look like. They won’t all be this easy!

Which of these two visualizations displays a coherent transition from left to right:

↑


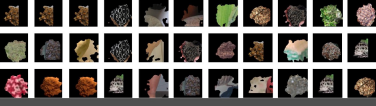
↓


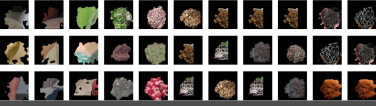
Respond using the up arrow (↑) to select the top visualization or the down arrow (↓) to select the bottom visualization. If you can’t tell, make your best guess.

Please take your time throughout the study. If you can’t tell which one is random, make your best guess.

Trial 25/288

Which of these two visualizations displays a coherent transition from left to right:

↑


↓


Respond using the up arrow (↑) to select the top visualization or the down arrow (↓) to select the bottom visualization. If you can’t tell, make your best guess.

Figure B.3: Full text instructions presented to participants of the user study.

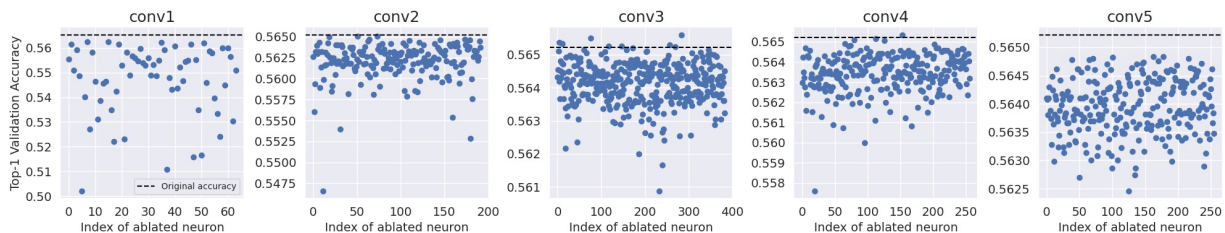


Figure B.4: For each of AlexNet’s five convolutional layers, we ablate neurons **individually** and measure the effect on ImageNet top-1 validation accuracy. Neurons were individually ablated by setting their activations to zero during the forward pass.

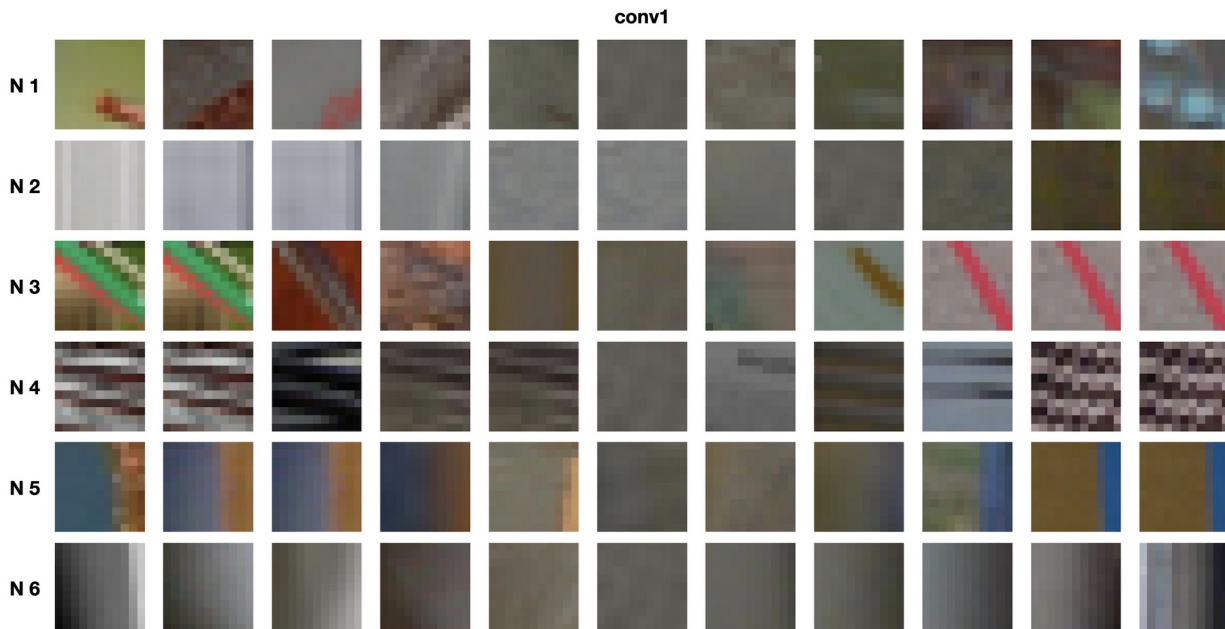


Figure B.5: Natural-RF visualizations along the first 6 neurons of conv1.

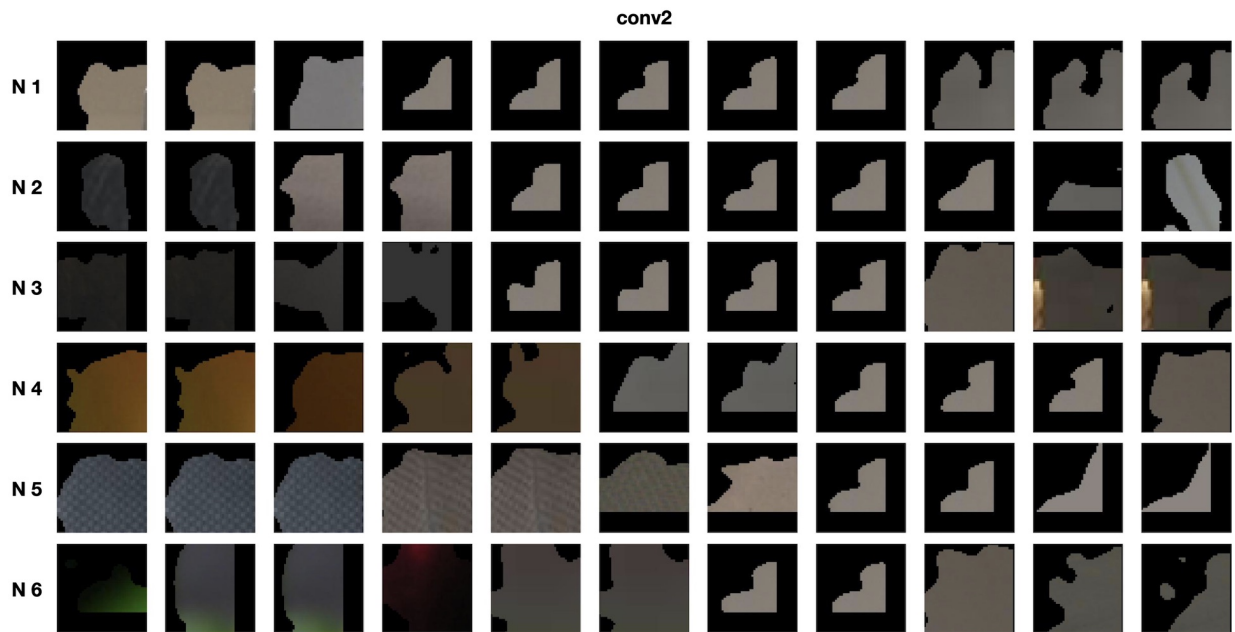


Figure B.6: Natural-RF visualizations along the first 6 neurons of conv2.

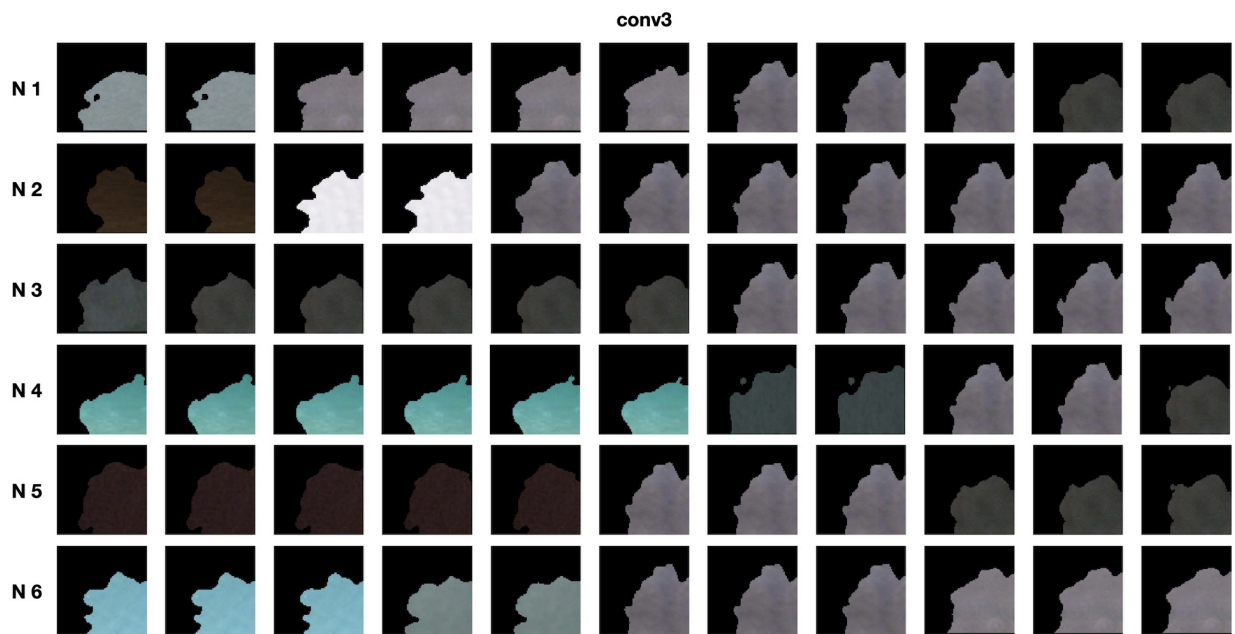


Figure B.7: Natural-RF visualizations along the first 6 neurons of conv3.

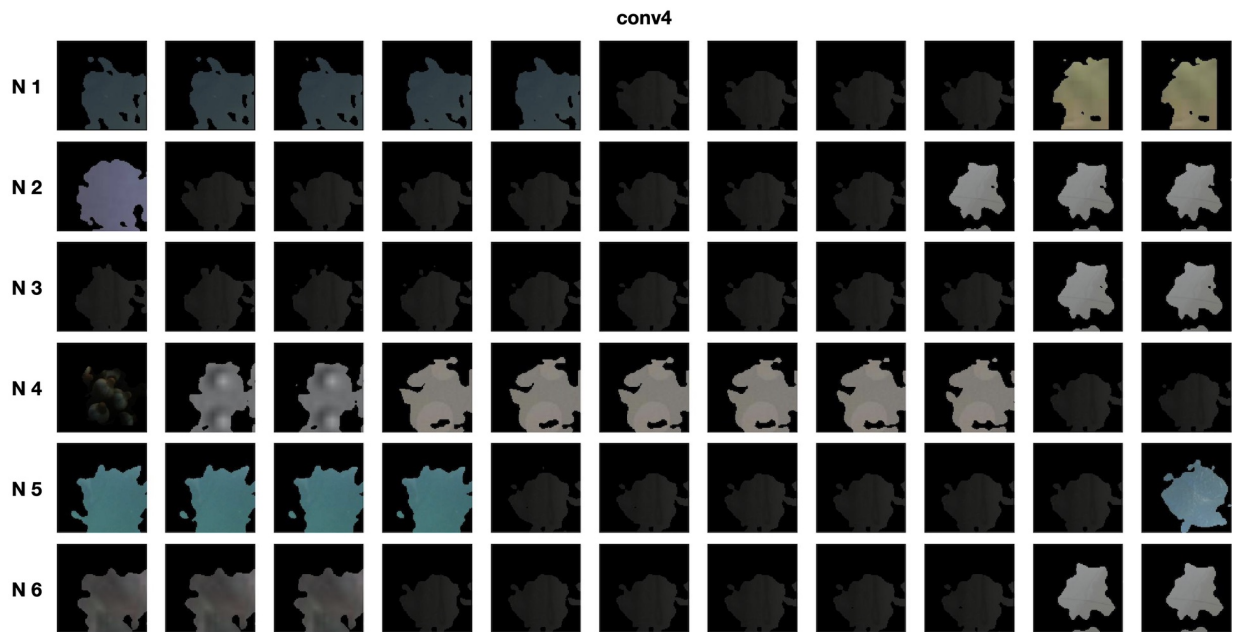


Figure B.8: Natural-RF visualizations along the first 6 neurons of conv4.

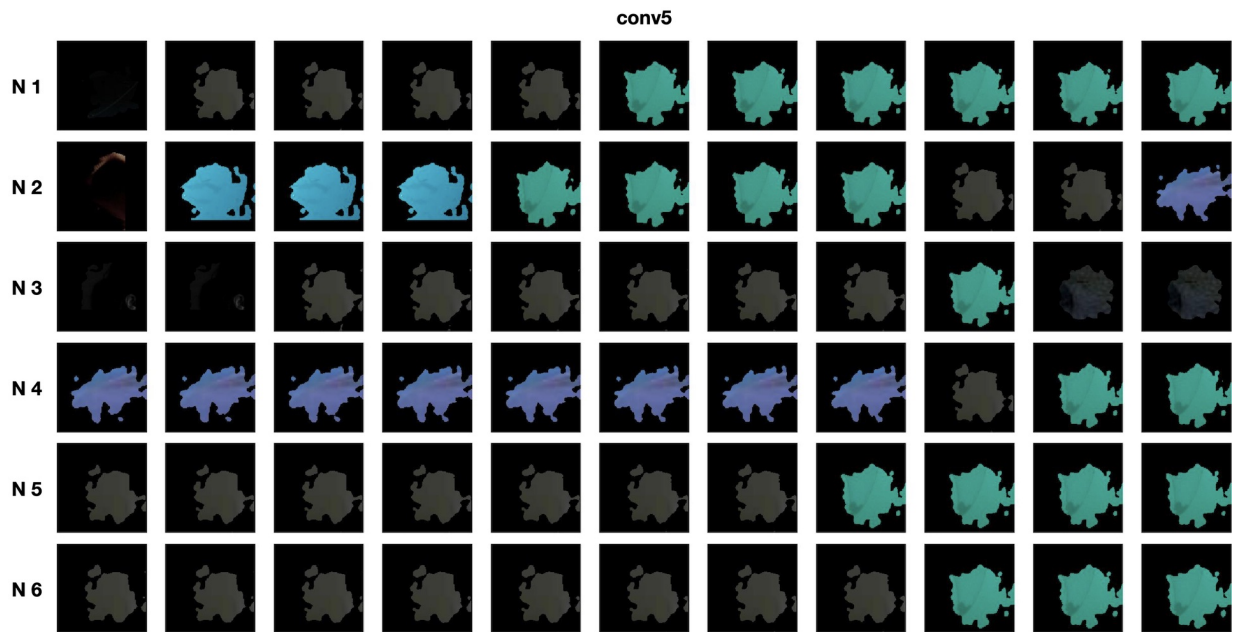


Figure B.9: Natural-RF visualizations along the first 6 neurons of conv5.

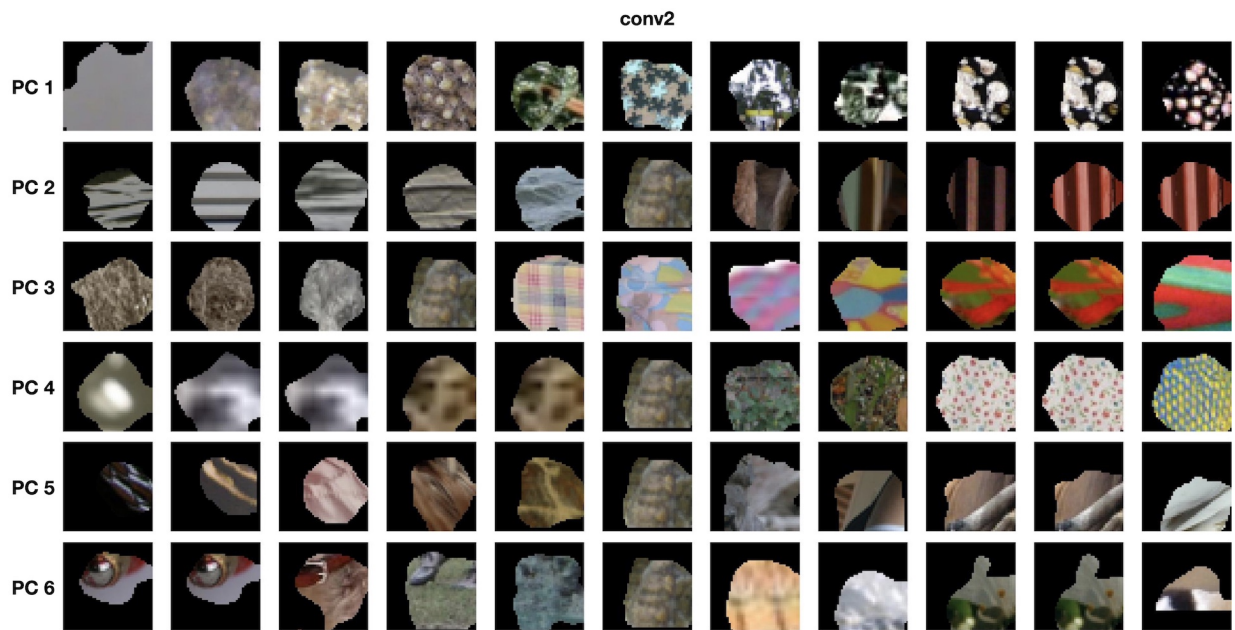


Figure B.10: Natural-RF visualizations along the first 6 principal components of conv2.

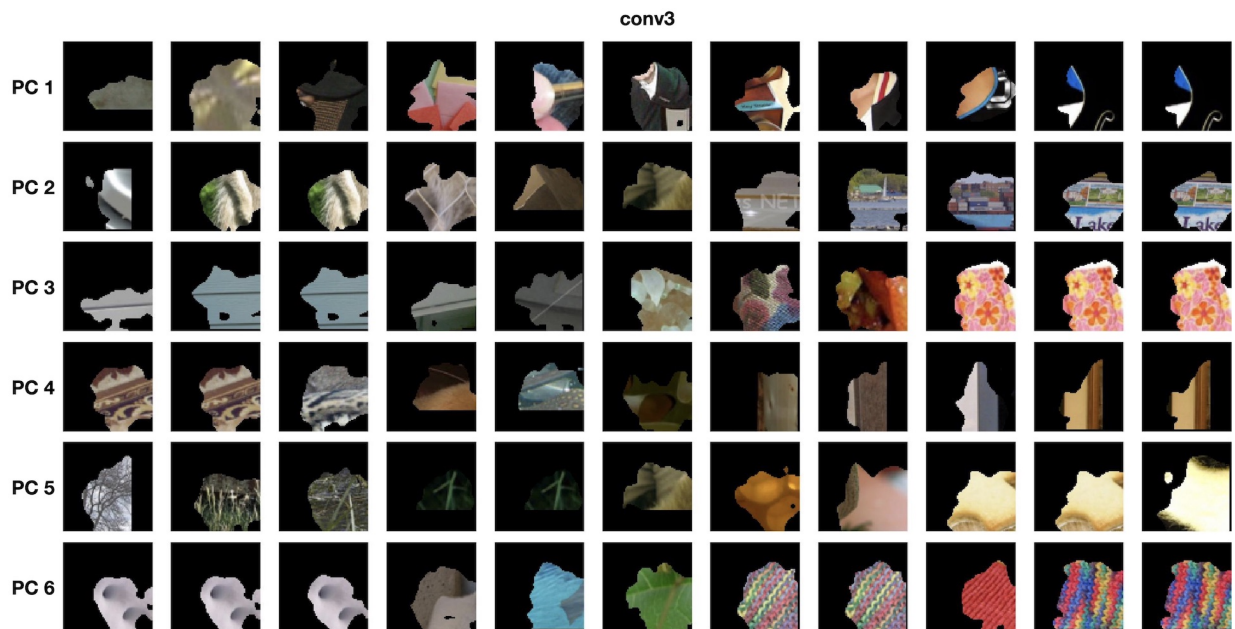


Figure B.11: Natural-RF visualizations along the first 6 principal components of conv3.

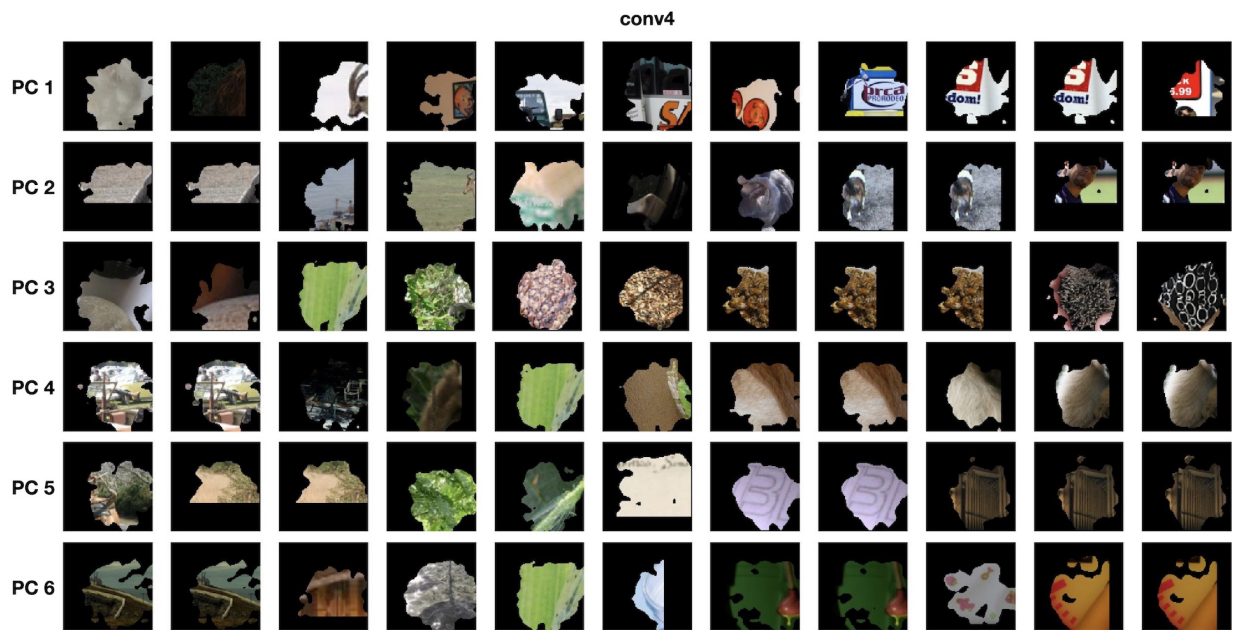


Figure B.12: Natural-RF visualizations along the first 6 principal components of conv4.

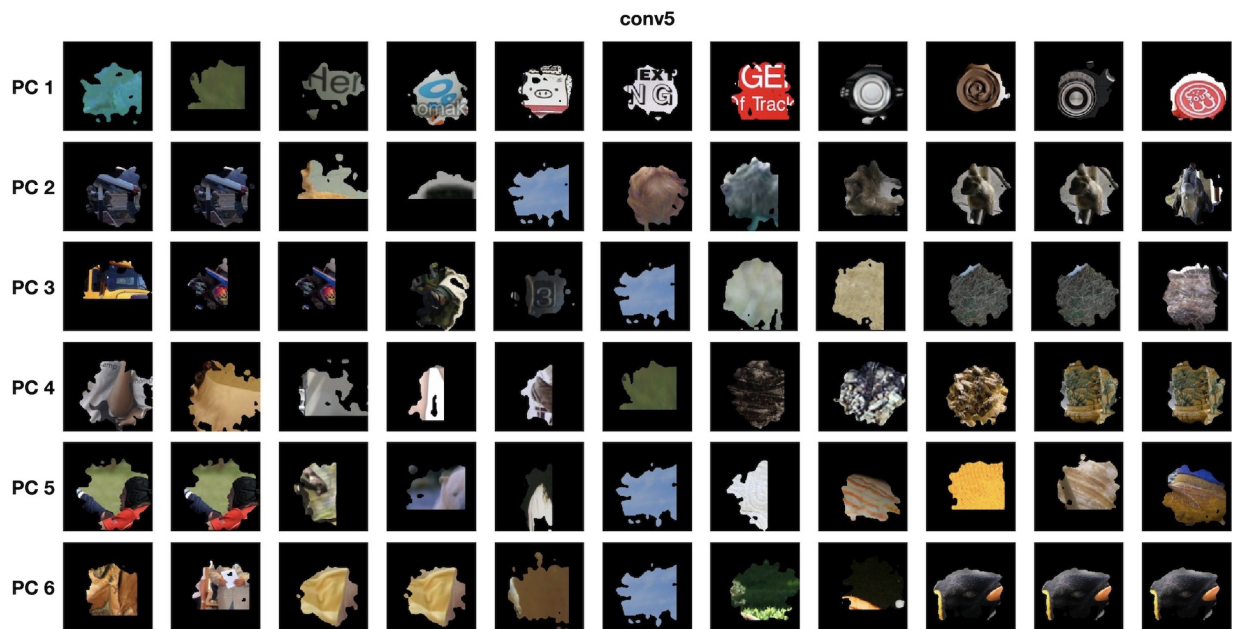


Figure B.13: Natural-RF visualizations along the first 6 principal components of conv5.

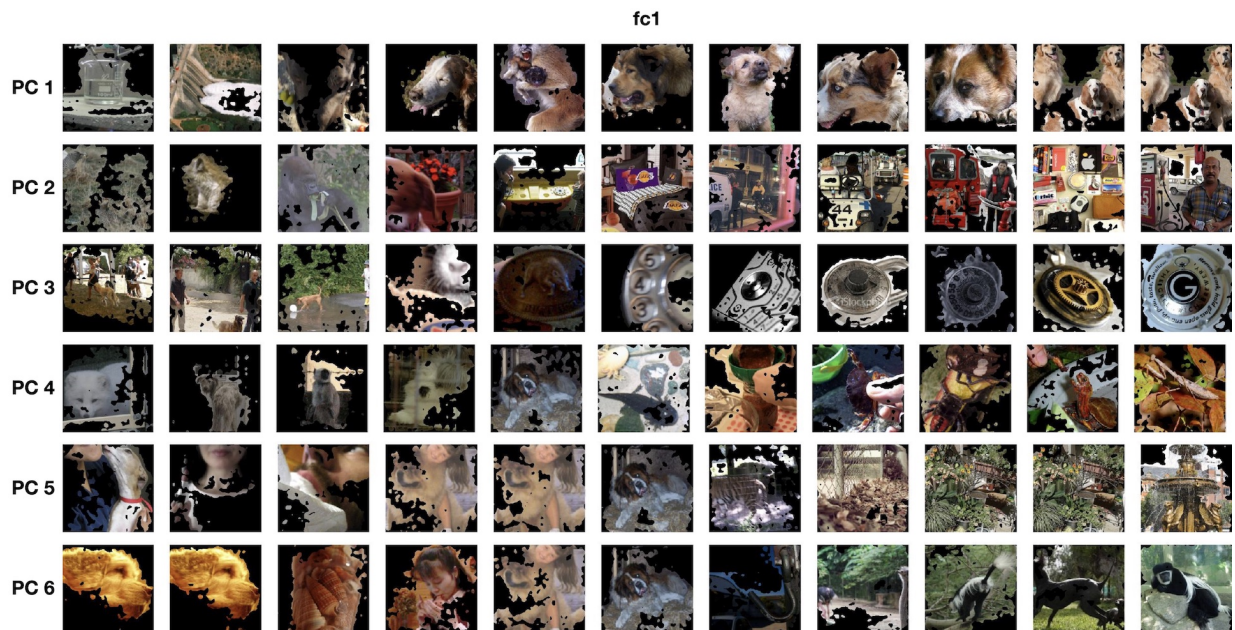


Figure B.14: Natural-RF visualizations along the first 6 principal components of fc1.

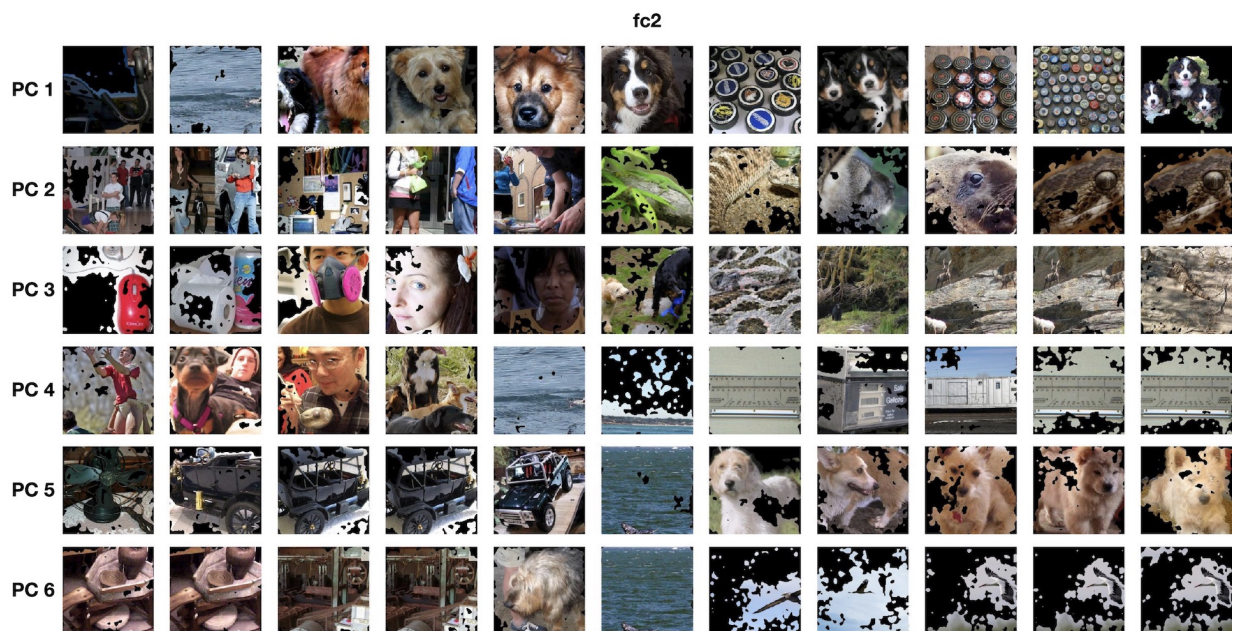


Figure B.15: Natural-RF visualizations along the first 6 principal components of fc2.

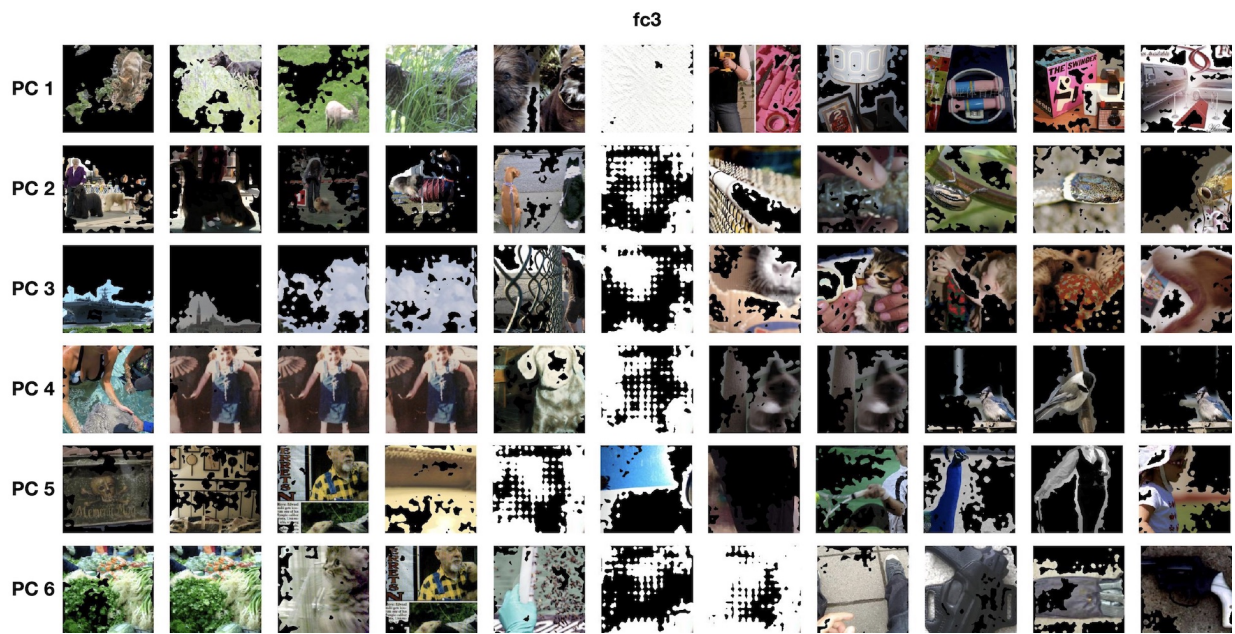


Figure B.16: Natural-RF visualizations along the first 6 principal components of fc3.