Mapping Visual Receptive Fields in Convolutional Neural Networks

Chin-Pu Fu

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Bioengineering

University of Washington

2023

Committee:

Wyeth Bair

Amy Orsborn

Program Authorized to Offer Degree

Bioengineering

University of Washington

Abstract

Mapping Visual Receptive Fields in Convolutional Neural Networks

Chin-Pu (Tony) Fu

Chair of Supervisory Committee:

Wyeth Bair

Biological Structure

Receptive field (RF) mapping is a fundamental technique for analyzing visual neurons. Neurophysiologists often use simple stimuli such as light spots or bars to locate visual RFs. While this approach is valid for early visual areas, where neurons are selective for simple features, its effectiveness in higher areas is less certain. The continued use of this technique for mapping RFs in higher areas may lead to misinterpretations of RF properties. With the advent of deep neural networks, we now have models that allow for the extraction of ground-truth RF properties. This methodological thesis focuses on developing and accurately quantifying RF mapping techniques in deep neural networks. Our findings indicate that RF mapping using bars is unreliable beyond early-to-mid layers, as bars do not drive neurons as effectively as natural images. This underscores the need for more robust RF mapping approaches for higher layers in deep neural networks but also for neurophysiology in general.

Table of Contents

Acknowledgement

I would like to express my gratitude to the Mary Gates Research Scholarship for providing financial support for this research. In addition, this work was supported by the National Institutes of Health NEI grant R01 EY029601 (W.B.) and NEI grant R01 EY027023 (W.B.). I am grateful for their generous support.

I would like to thank my PI, Professor Wyeth Bair, and a former member of Bair Lab, Dr. Dean Pospisil, for their guidance and support throughout my research journey. Special thanks to Professor Chris Neils for giving me the opportunity to work as his teaching assistant for the four quarters. His mentorship has been instrumental in strengthening my communication skills. I am also grateful to my co-advisor, Professor Amy Orsborn, for advising my thesis work.

I would like to acknowledge the Department of Bioengineering along with the faculty, staff, and fellow students for their support and encouragement.

Finally, I would like to thank Dr. Taekjun Kim and Dr. Anitha Pasupathy for hosting the CorticalNeurophys Journal Club along with Professor Bair and for their ongoing support and feedback on my research. Additionally, I am grateful to Sierra Schleufer and Luke Bun for organizing and leading the UW Vision Science Trainee Seminar Series, which has provided me with valuable opportunities to present my work and engage in discussions with other researchers.

Background

Receptive fields (RFs)

The receptive field (RF) of a neuron is defined as the sensory space in which a stimulus can modulate the response of that neuron (Sherrington, 1906). The sensory space depends on the type of neuron of interest. In the auditory system, for example, the RF is determined by the spectral characteristics of the acoustic stimuli. But more often, RF is spatial, like those first observed by Charles Sherrington in 1906. Sherrington coined the term RF when he described the scratch reflex of the skin (Sherrington, 1906).

In 1937, Haldan Hartline was the first to apply the concept of receptive fields (RF) in visual neuroscience (Hartline, 1938). Hartline removed the eyes of large bullfrogs (and previously horseshoe crabs) and recorded the action potentials of single optic nerve fibers while shining light onto the frog's retina. He observed that there were often fixed regions where he could excite particular optic nerves, and he roughly categorized them into three groups: (1) “on”: those that fired when there was light, (2) “off”: those that fired when light was removed , and (3) “on-off”: those that fired in response to a combination of the two. Hartline also noted that it was challenging to define the exact boundaries of the RF, as it highly depended on light intensity and whether the retina was adapted to the lighting conditions or not (Hartline, 1938). To gain a better understanding of the phenomenon, Stephen Kuffler, in 1953, studied retinal ganglion cells (RGCs) in cat's eyes. He discovered that the RF of these RGCs often consisted of a center region and a surrounding region, with the illumination of one region having the opposite effect from the other. Again, he noted the challenge of determining the extent of the RF (Kuffler, 1953).

During the same period, Horace Barlow conducted pivotal research on RFs in the retina (Barlow, 1953, 1961, 1964). Like Kuffler, Barlow revealed that retinal ganglion cells (RGCs) possess center-surround RFs (Barlow, 1953). Furthermore, Barlow demonstrated that many of these RGCs exhibit direction selectivity (Barlow, 1961, 1964). Subsequent research established that this selectivity results from interactions between bipolar cells, amacrine cells, and RGCs (Demb, 2007). While direction selectivity is an intriguing aspect of visual processing, this thesis will concentrate on visual spatial RFs, as most neural networks do not model temporal dynamics. Nonetheless, Barlow's influential work on efficient visual information coding has impacted the studies on these neural networks.

Delving deeper into visual processing beyond the retina, the groundbreaking work of David Hubel and Torsten Wiesel played a crucial role. Their four seminal papers (Hubel & Wiesel, 1959, 1962, 1963, 1965) significantly advanced our understanding of the functional organization and RF properties within the lateral geniculate nucleus (LGN), primary visual cortex (V1), and higher-order visual areas (V2 and V3, as explored in their 1965 paper). While

working at Kuffler's lab, Hubel and Wiesel projected shapes onto a screen in front of anesthetized cats, presenting various stimuli to gauge the location and properties of RFs. These stimuli included long, narrow rectangles (light or dark "bars") and straight-line borders of differing brightness (referred to as "edges"). They recorded single neuron responses in the cat's V1 area to these stimuli and identified neurons with preferred stimulus orientations (Hubel & Wiesel, 1959).

Hubel and Wiesel discovered that some orientation-selective neurons featured distinct "on" and "off" subregions within their RF (Hubel & Wiesel, 1959). When a light bar elicited significantly more action potentials than the baseline, they marked the light bar's location with X's, representing the "on" region. Typically, this elongated "on" region was bordered by two areas that would suppress the response if exposed to light, usually marked with triangles. Example mappings can be seen in **Figure 1**. The responses of these RFs were linear, meaning they resulted from a summation of the preferred stimuli presented to the subregions. Because of their simplicity, Hubel and Wiesel dubbed these neurons "simple cells" Hubel & Wiesel, 1959).

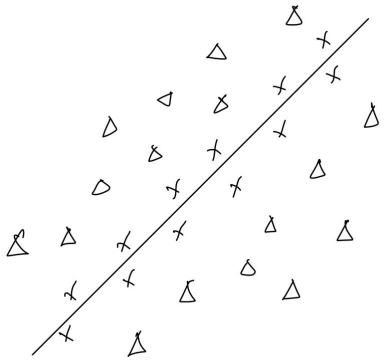

Figure 1. Example receptive field organization of cells in the cat's LGN and V1 areas (Hubel and Wiesel, 1961). X's indicate "on" regions, and triangles represent "off" regions. Simple cell with a preference for a light bar oriented at about 45°.

Besides simple cells, there are also "complex cells." These orientation-selective neurons do not have distinct subregions in their RFs, making it impossible to map them using X's and triangles (Hubel & Wiesel, 1962). Instead, Hubel and Wiesel typically outlined the area in which a stimulus could evoke a response, an approach we will also use in this thesis. Complex cell RFs are activated when stimuli of the correct orientation are present, making them selective for "lineness" rather than specific patterns of dark and bright regions.

Within the V1, V2, and V3 of cats and monkeys, Hubel and Wiesel also discovered cells that were sensitive not only to the orientation of the stimuli but also to their length (Hubel & Wiesel, 1965). These cells exhibited a weaker response when stimuli extended beyond a specific length, a property referred to as "end-stopping." This end-stopping characteristic is believed to contribute to the perception of corners and angles. Another related property of V1 is spatial frequency selectivity, which has been investigated by researchers such as Movshon et al. (1978) and De Valois et al. (1982) using sinusoidal gratings.

One of the challenges in RF mapping is the need for quantitative precision. Hand-mapped RFs are prone to human error, making more sophisticated techniques necessary for achieving greater quantitative power. Techniques such as spike-triggered averaging (de Boer & Kuyper, 1968) and reverse correlation (Jones & Palmer, 1987) have been employed to address this issue. Furthermore, as we progress into higher visual areas, neuronal selectivity becomes increasingly abstract. This requires ingenuity in designing complex stimuli that are also mathematically well-characterized. For example, Pasupathy and Connor (2001) developed a stimulus set consisting of 366 stimuli, characterized by curvature as a function of angular position. These stimuli were used to map the V4 area of macaque monkeys, showcasing the need for innovative approaches in RF mapping.

The history of RF mapping mirrors the progression of the visual pathway itself, advancing from basic localization and on-off organization to more complex shape selectivity, and ultimately, to object and face recognition. Over time, the methods for mapping RFs have become increasingly systematic and sophisticated. It is essential not to overlook this valuable knowledge when studying artificial neural networks (ANNs). As will be discussed in the remainder of this section, these two fields can mutually benefit from each other's insights. By leveraging ANNs, there is potential to further enhance the quantitative rigor of RF mapping techniques.

Convolutional neural networks (CNNs)

Hubel and Wiesel's contributions to science extend beyond the field of visual neuroscience. By connecting their findings in the LGN and V1 area to Kuffler's work, they gradually developed an understanding of the hierarchical organization of the visual pathway (Hubel & Wiesel, 1959). This understanding inspired Fukushima (1980) to propose the Neocognitron, a multilayered neural network that is widely regarded as the earliest model of convolutional neural networks (CNNs). Although neural networks already existed at this time, with researchers attempting to expand upon Rosenblatt's single-layer perceptron (Rosenblatt, 1962) through the introduction of multilayer structures (Ivakhnenko & Lapa, 1965), gradient descent (Amari, 1967), and backpropagation (Linnainmaa, 1970), many noticed that these artificial neural networks (ANNs) were sensitive to the position of the stimuli.

To incorporate spatial invariance into ANNs, Fukushima (1980) drew inspiration directly from Hubel and Wiesel's work (Hubel & Wiesel, 1962, 1965) and implemented a model consisting of alternating "S-cell" and "C-cell" layers. The "S-cell" layers, which were inspired by simple cells, detect local features such as edges. Meanwhile, the "C-cell" layers pool the responses from the S-cell layers, thereby enhancing shift and scale invariance. Over time, the S-cell and C-cell layers have evolved into the convolutional and pooling layers seen in today's CNNs (LeCun et al., 1998).

The term "convolutional" was first introduced by Yann LeCun and his colleagues at AT&T Bell Laboratories in 1989 to describe the operation being performed by their neural network (LeCun et al., 1989). They used a small convolutional neural network (CNN) with three hidden layers, known as LeNet-1, to recognize handwritten zip codes. LeCun replaced Fukushima's Hebbian learning rule with backpropagation and standardized certain aspects of the network architecture. Specifically, weight sharing, introduced by Rumelhart et al. (1986), became the core of the "convolution" operator. Instead of allowing different S-cells to be tiled across the spatial dimension, as in Neocognitron (Fukushima, 1980), the same kernel is strided across the space and performs dot-product operations with the input. However, it is worth noting that "convolution" is somewhat of a misnomer since the kernel is never flipped. (But the name “cross-correlational neural network” just does not have the same ring to it.)

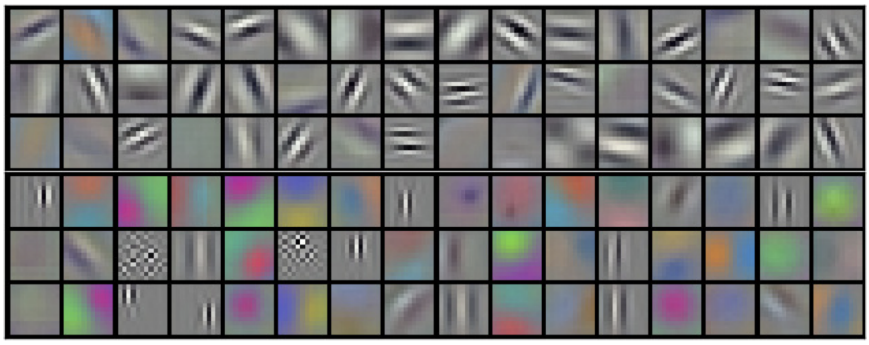

Figure 2. The 96 kernels in the first convolutional layer of AlexNet. Noticeable differences in coloration exist between the kernels of one GPU and the other, with one set being significantly more colorful (Krizhevsky et al., 2012). Note that the Pytorch version of AlexNet was trained on a single GPU, so while some kernels are achromatic and some are chromatic, the kernel’s id does not determine how chromatic it is.

In this thesis, we will conduct an analysis of three CNNs: AlexNet, VGG16, and ResNet18. Each of these networks represents a significant milestone in the development of deep learning. AlexNet won in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a considerable margin (Krizhevsky et al., 2012). Several factors contributed to its success. First, it was substantially "deeper" than its predecessors, consisting of five convolutional layers followed by three fully connected layers. As expected, training such a large neural network presented challenges. Alex Krizhevsky and his colleagues at Geoffrey Hinton's lab introduced several innovations to address these issues (Krizhevsky et al., 2012). They replaced the commonly used sigmoid activation function with the Rectified Linear Unit (ReLU) to mitigate the vanishing gradient problem, thereby increasing training speed and eliminating the need for normalization (although they still applied normalization after some ReLU layers). To enhance invariance and reduce overfitting, they employed data augmentation and "dropout," a technique that randomly "turns off" some neurons during training (Krizhevsky et al., 2012). Due to memory constraints, AlexNet had to be trained on two separate GPUs, which represented another breakthrough. Interestingly, the first convolutional layer (referred to as Conv1) exhibited functional segregation between the GPUs. One GPU contained more chromatic kernels, capturing color-specific information, while the other had achromatic kernels that functioned as

edge detectors invariant to color (**Figure 2**). There will be more discussion on the color selectivity in ANNs in the later part of this section.

Following the success of AlexNet, researchers continued to delve into deeper neural networks. In 2014, the Visual Geometry Group (VGG) from the University of Oxford developed VGGNets, a novel CNN architecture that employs small convolutional kernels to achieve remarkable depth with 11, 16, and 19 layer variations (Simonyan & Zisserman, 2014). For comparison, AlexNet has 8 layers. It will be interesting to study the RF of VGGNets and understand how it differs from CNNs using larger kernels.

VGGNets were soon surpassed by a much deeper network when Kaiming He and colleagues introduced the ResNet architecture in 2015 (He, Zhang, Ren, & Sun, 2015). They managed to create networks with depths ranging from 18 layers to an impressive 152 layers. This achievement was unprecedented since deep CNNs in the past often suffered from vanishing or exploding gradient problems. He et al. addressed this issue by incorporating identity mappings, allowing convolutional layers to learn residual differences and providing a "highway" for backpropagation. It will be interesting to study the impact of identity mapping on the RF of the CNNs.

Existing visualization methods

The primary advantage of studying Convolutional Neural Networks (CNNs) over biological visual systems is the accessibility of the learned parameters. For example, examining the kernels of AlexNet (**Figure 2**) reveals the features to which it is selective (Krizhevsky et al., 2012). This is possible because the kernel is dot-multiplied with the input, so the optimal input is always the kernel itself (up to a positive scaling factor). However, interpreting kernels beyond the first convolutional layer (Conv1) becomes challenging due to nonlinearities and the fact that their inputs are activation maps from the previous layer, rather than the RGB channel inputs that are human interpretable.

In non-Conv1 layers, a kernel has the same number of channels as the number of kernels in the previous layer. One approach to obtain a human-interpretable Conv2 kernel is to weight the Conv1 kernels by the Conv2 kernels, as demonstrated by Lee et al. (2008) when visualizing the second layer of a sparse variant of a Restricted Boltzmann Machine (RBM). We have used a similar method previously to investigate the mechanism behind border ownership selectivity in Conv2 of AlexNet (**Supplementary Figure 1**). Lee et al. (2009) extended the same method to visualize Conv3.

However, using the weighted combination of Conv1 kernels ignores the nonlinearities in ANNs, which plays a crucial role in building up complexity over layers and does so without canceling out integrating opposite features. For example, In **Figure 3** we showed that two images of reversed contrast resulting in almost identical activation maps because ReLU eliminates the negative responses.

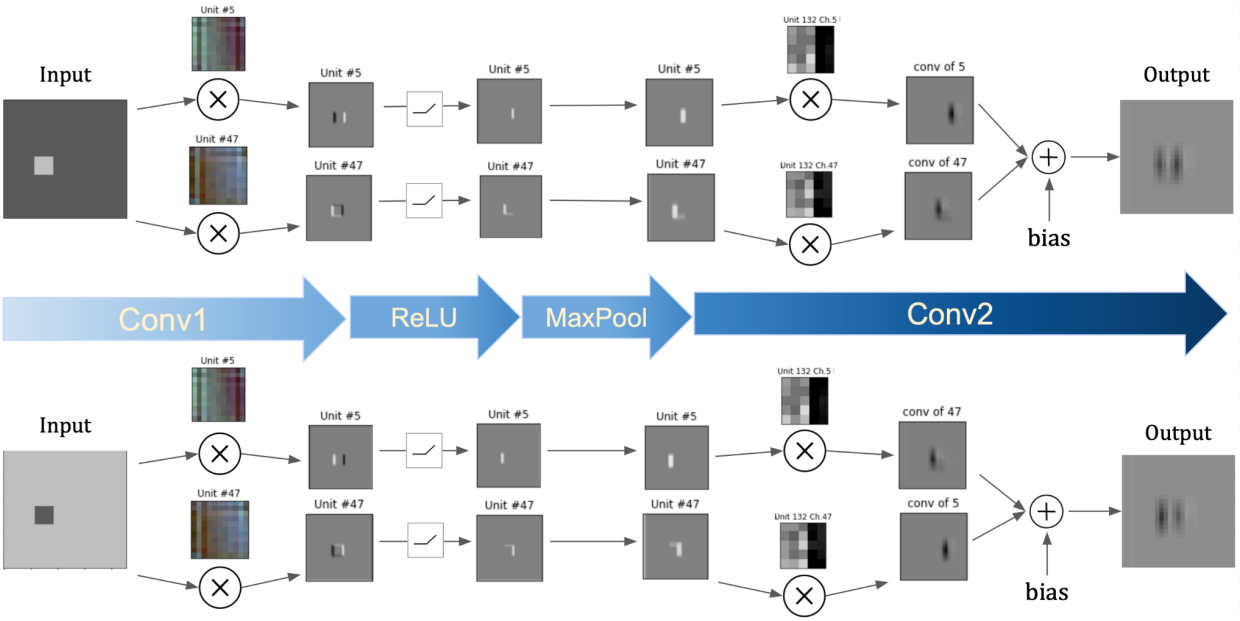

Figure 3. **Contrast Invariance of Conv2 Unit no. 132.** This figure explores the contrast invariance achieved by Conv2 Unit no. 132. The top and bottom rows display the processing of each input image through various stages in AlexNet up to Conv2. Despite having opposite contrast polarities, the input images produced nearly identical output feature maps after undergoing the same operations. To showcase the simplicity of this mechanism, only two Conv1 units (and consequently, two channels of Conv2 Unit no. 132) were used. The input images were first convolved separately with Conv1 Units no. 5 and no. 47, which are Gabor filters with opposing contrast preferences, highlighting the east and west edges of the square. The first ReLU layer removed negative responses, leaving the preferred edge of each unit. The first Max Pooling layer then sub-sampled the outputs of Conv1, which were separately convolved with Channels no. 5 and no. 47 of Conv2 Unit no. 132. Lastly, the feature maps of the two channels were summed to generate the final output. While the convolution by Conv2 was not essential for achieving contrast invariance, it contributed to left-sided border-ownership selectivity by making the left side of the edge inhibitory and the right side excitatory.

In 2009, Erhan and colleagues addressed this issue with a different approach called Activation Maximization (AM), an application of gradient ascent (Erhan et al., 2009). The method starts with an initial image, such as a uniformly distributed noisy image, and calculates the unit's response with respect to the input image to obtain the gradient. The image is then updated in the direction of the gradient. Gradient ascent is also employed in DeepDream, a technique developed by Google researchers Mordvintsev et al. (2015) to generate artistic images. The primary difference is that DeepDream often maximizes multiple units simultaneously and places a stronger focus on the artistic qualities of the resulting images, which may occasionally necessitate the use of additional smoothing techniques such as the regularization term in the Class Model Visualization introduced by Simonyan et al. (2013).

It is important to note that the optimization objectives of deep layers are highly nonlinear, which may cause gradient ascent methods to become trapped in local maxima. Recognizing the limitations of AM, Zeiler and Fergus (2013) at New York University proposed an alternative technique that offers researchers greater control over the generation of visualizations. The concept is simple: they constructed the same CNNs but in reverse. This network, called the Deconvolutional Neural Network (or DeconvNet), "feedforwards" the response unit/channel of

interest to earlier layers until reaching the input layer. The kernels are inverted, and the MaxPooling positions are stored using "switches" variables. Zeiler and Fergus also validated the DeconvNet visualizations by observing that the highlighted regions corresponded to areas where occlusion led to a significant drop in class prediction accuracy. Similar occlusion method was later used by Bolei Zhou and colleagues at MIT (2015). Zhou et al. occluded input images with random patches and compared the response to the original image. This allowed them to create a "discrepancy" map that indicated the importance of each region in the image to the unit.

It was demonstrated that Zeiler and Fergus's DeconvNet was almost mathematically equivalent to the backpropagation of the original networks (Simonyan, 2013). This led researchers like Simonyan et al. (2013) to wonder if it would be possible to visualize the gradient of the class score with respect to the input image. This "vanilla" backpropagation visualization was termed as the Saliency Map. However, these backpropagation visualizations often appeared noisy due to the inclusion of negative gradients.

To address this issue, Springenberg et al. (2015) introduced Guided Backpropagation, which modifies the backward pass of vanilla backpropagation in two ways. First, similar to DeconvNet, negative gradients are rectified during the backward pass. Second, the locations that result in negative values during the forward pass of the ReLU layer are also rectified. These modifications often result in cleaner gradient visualizations.

Bach et al. (2015) proposed a method that offers researchers more control over the relevance metric being back-propagated for visualization, with the aim of improving the fidelity of the generated explanations. Instead of backpropagating gradients, they backpropagate relevance scores calculated according to specific propagation rules, which are usually functions of the weights and the activations. One can choose the propagation rule based on the type of layer. This technique, known as Layer-wise Relevance Propagation (LRP), supports the standardization and generalization of propagation-based visualization techniques.

However, LRP can be somewhat complicated. For those who prefer the simplicity of linear combination, Zhou et al. (2016) introduced Class Activation Mapping (CAM), an intuitive visualization method that works as follows: After the final convolutional layer, they placed a global average pooling (GAP) layer, which collapses the spatial dimension and summarizes each channel into a single average value. This GAP layer is then followed by a fully connected layer. Based on the weights of the fully connected layer, the activation maps are produced as a linear combination of the feature maps immediately preceding the GAP layer. This concept can be generalized such that the visualization of the current unit is a weighted combination of previous activation maps. However, the CAM’s specific architectural constraints limit its application. Selvaraju et al. (2017) proposed an alternative method that eliminates the need for the GAP layer. They suggested performing average pooling on the gradients of the classification output with respect to each spatial position of the convolutional layer. They named this method Grad-CAM, which also improves spatial resolution of the visualization by fusing it with guided backprop through pointwise multiplication with Guided Backprop.

Unfortunately, we could not cover every visualization technique in this section. In the thesis, we employed guided backpropagation, gradient ascent, and occlusion method. For more information of the implementations of the other visualization techniques, we recommend exploring these resources:

- Pytorch's Captum library
- GitHub repository by utkuozbulak: https://github.com/utkuozbulak/pytorch-cnn-visualizations#gradient-visualization
- GitHub repository by hans66hsu: https://github.com/hans66hsu/nn_interpretability

CNNs as tools to study biological visual systems

The concept of using CNNs as tools to investigate biological visual systems can be traced back to the early days of CNNs. Fukushima (1980), when discussing Neocognitron, also expressed the difficulty in studying how animals achieve pattern recognition. He stated, "If we could create a neural network model with the same capability for pattern recognition as a human being, it would offer us a powerful clue to understanding the neural mechanism in the brain" (Fukushima, 1980). It is fair to say that today's CNNs possess object recognition abilities that are comparable to, or even surpass, human performance.

It has been observed that the first layer of these CNNs exhibits Gabor-like filters, which resemble the simple cells found in the V1 area of the visual cortex (Hubel & Wiesel, 1959). Additionally, these Conv1 kernels are similar to the basis functions learned by Olshausen and Field's (1997) computational model for sparse coding of natural images. Color selectivity in CNNs has also been studied. Recall that Conv1 kernels of AlexNet were either distinctly chromatic or achromatic (Krizhevsky et al., 2012). It has been demonstrated that the chromatic kernels demonstrate color opponency, which is a characteristic observed in biological visual systems (Flachot & Gegenfurtner, 2018).

The correspondence between CNNs and biological visual systems can also be observed in V2/Conv2 layers. Recall that Lee et al. (2008, 2009) used weighted combinations of Conv1 input to visualize kernels of deeper layers. Their visualizations were employed to compare angle profiles from Ito and Komatsu's (2004) empirical study, which assessed how well V2 neurons responded to a set of angle stimuli. They identified corner and junction detectors in Conv2, which increased confidence in the potential for correspondence between the two systems at higher levels.

In fact, most studies that aim to use CNNs as a new framework for examining biological visual systems typically begin by comparing the response profiles (to a common set of stimuli) of both systems. For instance, in order to investigate if CNNs possess units with translation-invariant boundary curvature selectivity, similar to those found in primate V4, Bair Lab compared the angular position and curvature (APC) tuning profile of the primate V4 area and pretrained AlexNet (Pospisil et al., 2018). In my undergraduate thesis, we attempted to

identify border ownership units in AlexNet. Our approach also began by comparing the response of AlexNet units to Zhou et al.'s (2000) border ownership stimuli. A description of our method can be found in **Supplementary Figure 2**.

Other than electrophysiological data, researchers have also used functional MRI (fMRI) signals recordings to compare CNN activity with biological neural systems. Khaligh-Razavi and Kriegeskorte (2014) compared 37 computational models and discovered that AlexNet could represent inferior cortex representation. In an effort to determine which CNN layers correspond to the selectivity of the ventral pathway, Güçlü and van Gerven (2015) recorded functional fMRI signals from human subjects presented with images from various categories. They related voxels that responded similarly to the same set of images with layers of a modified version of the trained AlexNet (created by Chatfield et al., 2014). By truncating the model up to the layer of interest, they trained an additional model that linearly combined the truncated model responses to predict the fMRI signals. The models were successful in predicting the fMRI signals of humans, albeit with decreasing accuracy as they delved deeper into the layers.

While CNNs provide an exciting new direction for understanding the brain, it is crucial to approach the interpretation of their results with caution. As Crick (1989) pointed out, the backpropagation algorithm is not biologically plausible because neurons typically do not propagate weight information upstream.

Knowledge gap

RF mapping, specifically determining the center and size of an RF in the visual field, has been an important but often hazy step in the neurophysiological characterization of visual neurons. Uncertainty arises when only a very limited set of stimuli may be shown because of time constraints and technological limitations, and importantly there is no ground truth reference for comparison. Mapping the spatial location and extent of RFs is critically important in many experiments to avoid erroneous results. In the study of visual motion sensitivity, Barlow et al. (1964) warned, "... it is easy to be misled into thinking a unit is directionally selective if one tests its responses to movement before mapping out the receptive field." Indeed, a neuron with no direction bias can be determined to be highly direction selective if stimuli are not properly centered on the RF (Barlow et al., 1964, their Fig 4).

In the study of modulatory (a.k.a., "non-classical") RF regions that surround the classical receptive field (CRF), Walker et al. (2000) stated, "Errors in centering or sizing of the CRF can result in misleading conclusions about the relationship between the center and the surround." Indeed, whether the RF surround is determined to be suppressive or facilitatory can hinge on how the RF size was determined (Cavanaugh et al., 2002). Equally serious misinterpretations can occur in studies of orientation tuning and translation invariance (Pasupathy et al., 2018) when stimuli are misaligned to the RF.

Here, we take advantage of ANNs to provide hierarchical visual systems of moderate sophistication where approximate ground-truth estimates of RF maps can be computed. We use these to understand when and how simple RF mapping paradigms may break down.

Methods

After preparing the animal and inserting the electrode, an electrophysiologist approached a visual neuron by first estimating the general location of the RF using simple visual stimuli, such as a spot of light or a bar. The stimulus is moved around until it evokes a response from the neuron. The electrophysiologist then focuses on that area and presents a more diverse set of stimuli to refine the RF boundaries before starting to characterize the neuron's RF properties.

With CNNs, RF mapping can be automated with high throughput. In this section, we explore the application of neurophysiology-inspired techniques to delineate the effective receptive field (ERF) of CNNs. ERF is distinct from the maximum receptive field (MRF), which can be easily determined through calculations involving stride, kernel size, padding, and/or dilation of the layers.

Our goal is not to characterize the RF properties, but to assess the effectiveness of the neurophysiology-inspired techniques in outlining the ERF. To assess their performance, we will compare the ERFs obtained through them with those derived from "ground-truth" methods. These ground-truth approaches involve evaluating a large set of natural images to identify optimal stimuli, as well as leveraging the internal parameters or gradients to accurately map the ERF. Additionally, we will discuss the metrics we used to assess the "mappability" of the units as well the metrics used to explain the mappability..

Lastly, we have developed a web app for real-time RF mapping of CNNs, which allows users to visualize and interact with the receptive fields directly through their browsers. For more detailed information on the implementation of these methods, please refer to our GitHub repository: https://gitfront.io/r/tonyfu97/SK2aBW5ZfHvh/borderownership/.

Neurophysiologically inspired RF mapping methods

In this section, we have explored various neurophysiology-inspired stimuli, such as achromatic bars, color bars, and sinusoidal gratings. We will discuss the advantages and disadvantages of each stimulus type, and provide their respective pseudocode implementations in the Appendix to ensure reproducibility.

Achromatic bars

We presented bars using pixel values of 1 (R, G and B) for white and -1 (R, G and B) for black. Our achromatic set included white bars on a black background, and black bars on a white background. In total, the achromatic bar set consists of 33984 unique bars with different bar lengths (24/32, 12/32, 6/32, and 3/32 M, where M is the side length of the MRF), aspect ratios

(0.5, 0.2, and 0.1), 8 orientations (0°, 22.5°,... 157.5°), 2 contrasts, and positions that are evenly distributed the cover the entire MRF (**Figure 4**).

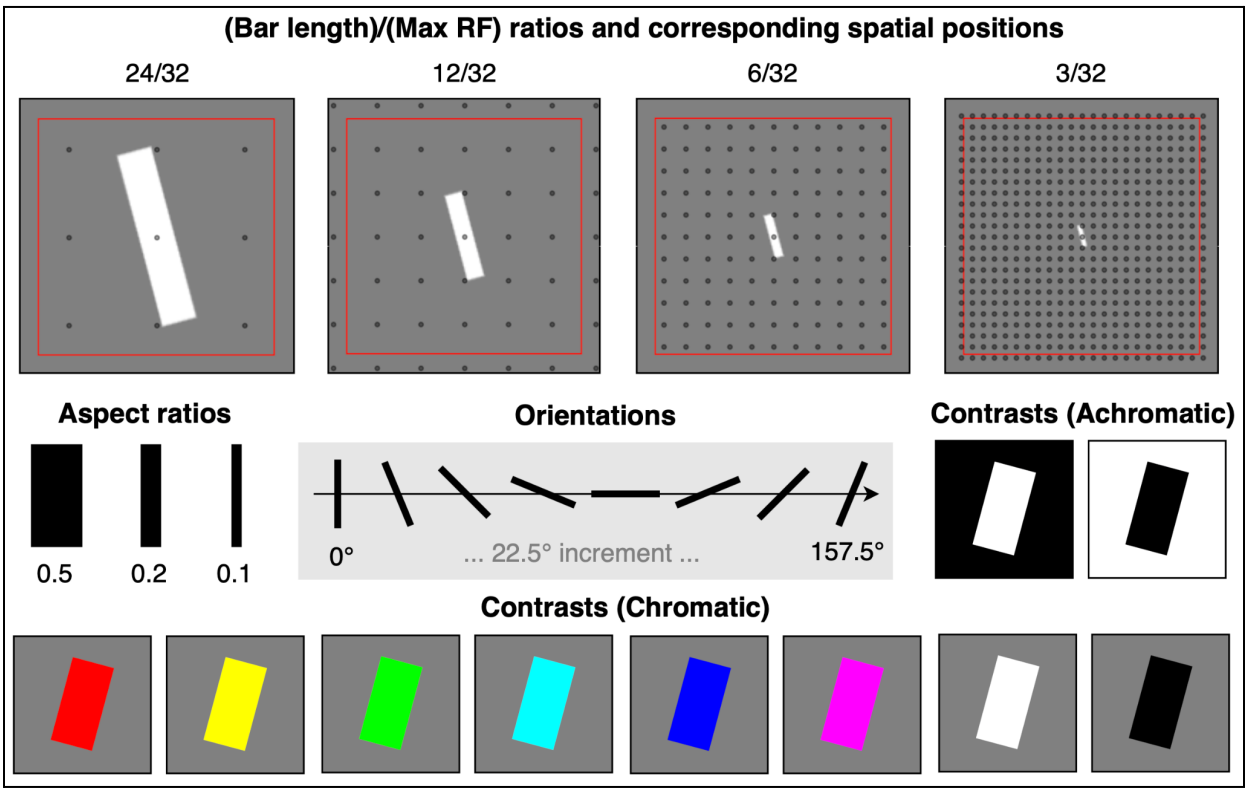

Figure 4. Bar Stimulus Set Parameters. Our bar stimuli were defined using four parameters. (1) Bar Length: Four different lengths were used, expressed as fractions of the Maximum Receptive Field (MRF) size (i.e., 24/32, 12/32, 6/32, and 3/32). The actual length of each bar is determined by multiplying the MRF size by the corresponding ratio. The length of the bar also influences its spatial distribution within the MRF; shorter bars result in denser distributions. (2) Aspect Ratio: Bars were designed with one of three aspect ratios: 0.5, 0.2, and 0.1. (3) Orientation: Each bar could have one of eight possible orientations, ranging from 0 to 157.5 degrees, incremented by 22.5 degrees. (4) Contrast: Achromatic bars were presented with two contrast levels, while color bars had eight contrast levels. These four parameters combine to create a diverse set of bar stimuli.

When the MRF is too small ($M \leq 21$ pixels), the placements of bars become inconsistent with the deeper layers, and many small bars are unrecognizable. As a result, we excluded layers with small MRFs (AlexNet Conv1, VGG16 Conv1-4, and ResNet18 Conv1-2) from our analysis.

When creating the weighted bar maps, we prevent bars from canceling each other out by not using negative pixel values. This is achieved by always adding a white bar (weighted by the response value) on a background of zeros, even when the corresponding bar is black (**Figure 5**). Consequently, our maps do not convey contrast or color selectivity; they simply outline the region to which the unit is sensitive. To determine the appropriate number of bars to add, we looked at empirical evidence from AlexNet (**Supplementary Figure 3**), which suggested that a response threshold of 0.4-0.7 yields a bar map most similar to the ground truth map. Consequently, we chose a response threshold of 0.5. This means that all bars resulting in a

response below the cutoff response will be discarded. As a result, some units will have bar maps composed of thousands of bars, while others will have no bars at all (if their max response is negative). However, on average, the number of bars included at a threshold of 0.5 is in the order of hundreds, as shown in **Supplementary Figure 4**.

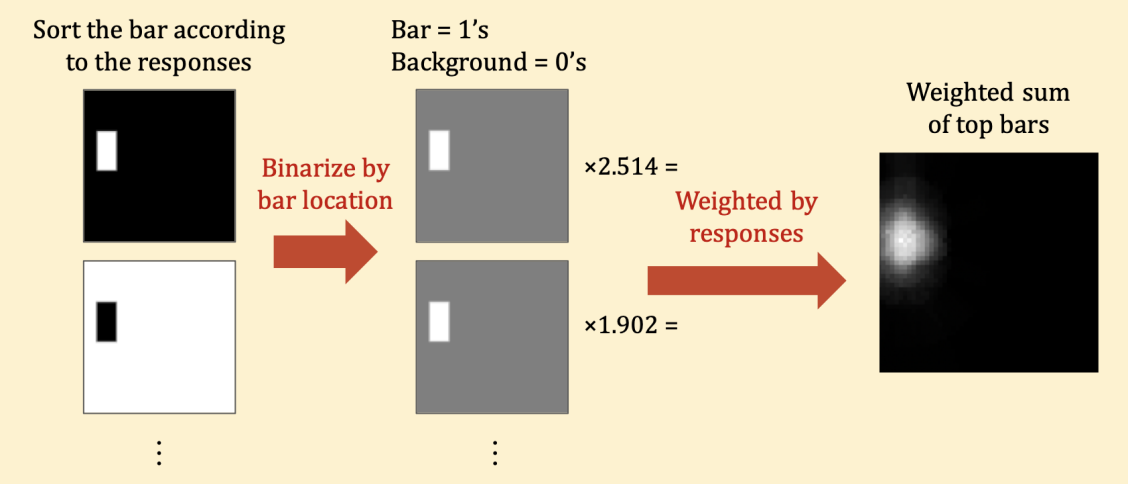

Figure 5. Making of a bar map. After presenting all 33,984 achromatic bars to the unit, the bars are ranked based on their elicited responses. Bars with responses exceeding the cutoff response are binarized, such that bar locations are assigned a value of 1 and background pixels are assigned a value of 0. The resulting binarized location maps are then weighted and combined to generate the final bar map.

However, the ERF mapped using this method does not account for suppressing regions that yield negative responses. Therefore, we had to repeat the same steps to create bar maps with bars that generate the most negative responses, in order to identify regions that can be suppressed by these bars. The cutoff responses have the following formulas:

```
R_cutoff_top    = max(RESPONSE_THRESHOLD * R_max, 0)        Eq.1
R_cutoff_bottom = min(RESPONSE_THRESHOLD * R_min, 0)        Eq.2
```

Advantages:

- Unlike some other RF mapping methods, which require the stimuli to be tailored for each individual unit, this approach allows all bars to be presented to all units of a given layer at the same time. This makes the maps fast to compute.
- Using achromatic bars is a standard practice in neurophysiological RF mapping, and this approach allows us to establish connections with traditional RF mapping techniques.

Disadvantages:

- It is likely that only a small portion of the bar is responsible for the unit's response. However, when creating the bar map, the entire bar is added, which could potentially lead to an overestimation of the size of the ERF.

- Although weighting the bars and summing them up is an intuitive solution, the bars are not orthogonal, meaning different bars may contain the same "interesting" part that drives the unit. By using the weighted sum, we are implicitly assuming that the RF property is linear. For example, if `bar1` and `bar2` result in responses of `2.514` and `1.902`, respectively, the bar map would be calculated as `2.514 × bar1 + 1.902 × bar2` (**Figure 5**). This assumption is unlikely to be accurate given the nonlinearity of CNNs. However, this may not be a significant concern, as we are not characterizing the RF properties, but simply aiming to outline the ERF.
- As the number of bars varies depending on the units, some units requiring thousands of bars can become a "bottleneck" for this approach when the bar map generation process is parallelized.
- Numerous units are not activated by achromatic bars. As will be discussed later, this could be due to bars lacking visual richness compared to natural images, and because some units exhibit strong color selectivity. This may lead to inaccurate mapping.

Color bars

The color bar set consists of 339,84 × 4 =135,936 bars, as it includes six additional colors (red, green, blue, yellow, cyan, and magenta) besides black and white. However, incorporating color adds complexity when interpreting the maps. For example, when presenting a magenta bar on a green background to the network, it is unclear whether the responses should be attributed to the bar, the background, or both. To create bar maps, the background must remain neutral so that most responses can be attributed to the bars. As a result, all background pixels are set to zero (gray), effectively "turning off" the Conv1 weights at those pixels by multiplying them with zero. The color values for bars were shown in **Table 1**.

Table 1. RGB values of bar colors.

Bar color	R	G	B
black	-1	-1	-1
white	1	1	1
red	1	-1	-1
green	-1	1	-1
blue	-1	-1	1
yellow	1	1	-1
cyan	-1	1	1
magenta	1	-1	1

Again, to prevent bars from canceling each other out, we modify the pixel values when creating the bar maps. In this case, we rectify the RGB channels so that negative RGB values become zeros. For example, if the bar is red (on a gray background), red is represented as (1, 0, 0) when creating the bar map. The bars are then weighted by the responses before being added to

the map. The cutoff responses are the same as for achromatic bar maps, as defined by **Eq. 1** and **Eq. 2**.

Advantages:

- Just like achromatic bar maps, all color bars can be presented to all units of a given layer simultaneously, making it a fast method.
- Color bars are more optimal for color selective units.

Disadvantages:

- Just like the achromatic bar maps, it is possible that only a small portion of the bar is responsible for the unit's response. We are also assuming that the RF is linear by doing a weighted sum. The number of bars included in the bar maps varies with the unit, leading to bottlenecks.
- Just like the achromatic bar maps, it is challenging to differentiate between regions that are suppressed by the presence of a bar and regions that are not modulatory at all using only the top bar maps (i.e., the bar maps created using bars that yield responses above the cutoff response given by **Eq. 1**).

Sinusoidal gratings

The sinusoidal grating is achromatic, and its pixel values vary from -1.0 to 1.0. It is presented as a circular patch on a gray background (of zeros). In total, the sinusoidal grating set consists of 113280 unique gratings with different diameters (24/32, 12/32, 6/32, and 3/32 M, where M is the side length of the MRF), spatial frequencies (0.02, 0.04, 0.08, 0.16, and 0.32), 4 spatial phases (0, 90, 180, 270 degrees), 8 orientations (0°, 22.5°,... 157.5°), and positions that are evenly distributed the cover the entire MRF (**Figure 6**).

Disadvantages:

- Typically, sinusoidal gratings are used to characterize the RF's spatial frequency selectivity once the RF has already been located. It is not surprising that we discovered that using sinusoidal gratings to map the ERF is not ideal, as the resulting map often consists of large overlapping circles that come from the large circular patches of the gratings. It is also frequently evident that only a portion of the patch drives the unit. Consequently, using sinusoidal gratings may significantly overestimate the ERF size.

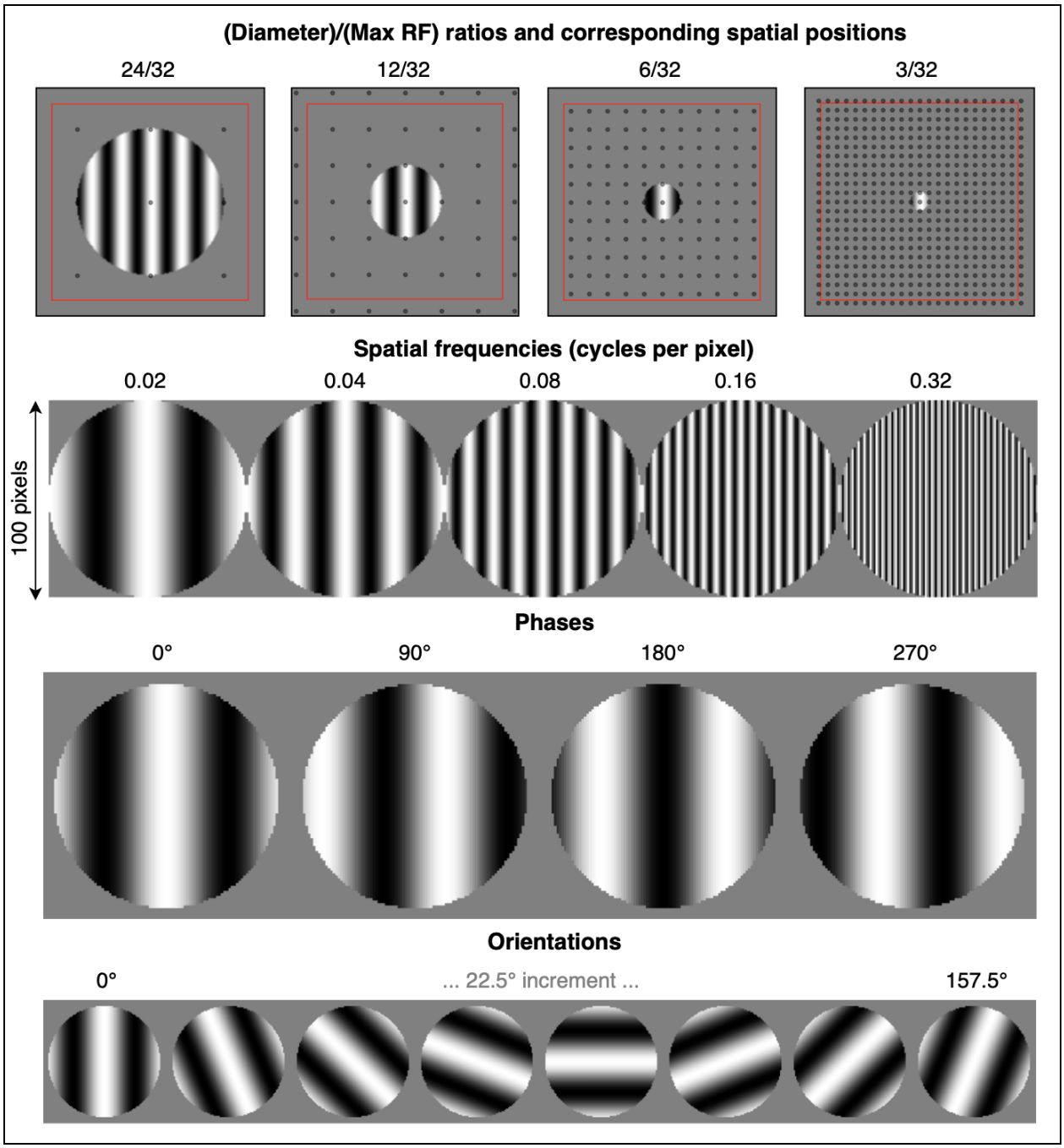

Figure 6 Sinusoidal Grating Set Parameters. Sinusoidal gratings are presented in circular patches and parameterized by four factors: (1) Diameter: Four different diameters were employed, expressed as fractions of the Maximum Receptive Field (MRF) size. The grating patch diameter also dictates its spatial placement within the MRF; smaller patches result in denser distributions. (2) Spatial frequency: Five possible spatial frequencies were used. (3) Phase: Each grating could have one of four possible phases, ranging from 0 to 270 degrees in 90-degree increments. (4) Orientation: Each grating could have one of eight possible orientations, ranging from 0 to 157.5 degrees in 22.5-degree increments.

“Ground truth” methods

As mentioned previously, there are several benefits for studying ANNs. One such advantage is that ANNs are deterministic, and their kernel weights are accessible. Moreover, PyTorch's automatic differentiation engine, Autograd, enables recording of input tensor operations and gradient computation (It should be noted most visualization techniques covered in the Background section dated prior to the release of PyTorch, which came out in late 2016). These features make it possible to extract information from ANNs that is not possible when studying the brain. To evaluate our neurophysiologically- inspired RF mapping methods, we employed four "ground-truth" methods. These methods rely partly on knowledge of the network's architecture (weights and connections) and internal responses along the pathway from the image to the unit under study, providing a reference point for evaluation. Below, we have summarized the methods we used, including their advantages and disadvantages.

Natural images

We used a set of 50,000 natural images from the ImageNet test set, which is also used in the Pospisil (2018) paper. These images are available on our lab website [http://wartburg.biostr.washington.edu/loc/course/artiphys/data/i50k.html]. The images were preprocessed into NumPy arrays with a shape of 3 × 227 × 227, and the RGB channels of the 50,000 images were batch normalized separately to ensure each channel has an average mean of zero and an average standard deviation of one.

Our objective is to find the image patches that generate the most positive response for every unit. The size of these patches is determined by the size of the MRF of the unit. For the sake of brevity, we will refer to these patches as "top patches". Since convolutional operation involves tiling the same kernel across the spatial dimension, creating multiple spatial instances of the same unit, it is computationally expensive to compare the responses corresponding to all patches of all 50,000 images. To mitigate this issue, we assume that there can only be one possible top patch in each image. Therefore, we first identify the top patch of an image and then rank the top patches of the 50,000 images to identify the top 100 patches.

Advantages:

- From experience, natural images are often our first choice for visualizations when studying neural network visual selectivity, because they are readily understandable to human observers.
- Natural images contain rich visual cues. Those complex and nuanced features may not be easily recapitulated with simple shapes or noises.

- Natural images can serve as the basis for other visualization techniques, such as guided backpropagation and gradient ascent.

Disadvantages:

- Natural images by themselves cannot be used to outline the RF, since we do not know which part of the image patches are driving the unit.
- Natural images are more difficult to characterize mathematically compared to simple shapes, making it harder to quantify the selectivity of neural network units. The complexity of natural images also makes it harder to isolate the specific properties that are driving the unit's responses, potentially leading to ambiguous or misleading results.
- The dataset used to study neural network visual selectivity may not contain the optimal images for certain units, limiting the scope of analysis.

Guided backpropagation

We used Utku Ozbulak's implementation of guided backpropagation, which is available on GitHub [https://github.com/utkuozbulak/pytorch-cnn-visualizations]. We made slight modifications to this implementation to compute unit-specific gradients instead of class-specific gradients. Since each top patch has a corresponding spatially offset copy of a unit, this algorithm only backpropagated the corresponding spatial unit to compute its gradient with respect to the input image. By applying a half-wave rectification to gradients based on the signs of both the forward and backward outputs, the guided backpropagation algorithm ensured that only the pixels with positive contributions are retained in the final visualizations.

To visualize the contributions of different regions of the top patch to the unit's output, we used guided backpropagation to generate visualizations for the top 100 patches. We then averaged the RGB channels then took the absolute value of the pixel values. This computes a 2D map that locates the areas in the image that can modulate the response. The brightness of each pixel in the map indicates the strength of the response, with brighter pixels indicating stronger responses. We then averaged the 100 2D maps to produce a single 2D visualization that summarizes the collective modulatory effects of the top patches.

Advantages:

- Guided backpropagation augments the top patch visualizations by highlighting the locations in an image that contribute positively to a unit's responses.
- Guided backpropagation is relatively fast, allowing for the incorporation of multiple top patches (100 in this case). It is important to include multiple images because, as Springenberg et al. (2015) pointed out, selectivities of deeper layers are more invariant, and it is unlikely a single image can fully capture the invariance.

Disadvantages:

- Like many gradient-based attribution methods, guided backpropagation can produce noisy and hard-to-interpret visualizations.
- It is important to note that guided backpropagation removes negative gradients at each ReLU layer, thereby producing a "cleaner" visualization. While it makes sense to set pixels to zero if they produce a negative response during forward propagation, as they would be rectified to zero anyways, it is more difficult to understand why negative gradients should be removed during backward pass. Zeiler and Fergus (2013) did not provide justification for this approach. It's worth noting that by removing the negative gradients at each ReLU layer, there is a risk of underestimating the size of the ERF.

Gradient ascent

Gradient ascent is a fundamental technique used in the visualization methods Activation Maximization (Erhan et al., 2009) and DeepDream (Mordvintsev et al., 2015), which generates artistic and psychedelic images specific to a channel of a layer. Unlike gradient descent, which focuses on minimizing the loss function by adjusting the parameters, gradient ascent aims to maximize the response of a specific channel by iteratively modifying an input image. To achieve this, it computes the derivative of the response with respect to the input image and takes a small step (with a learning rate of 0.1) in the direction of the locally steepest ascent. By repeating this process iteratively, it attempts to generate an image that elicits a stronger response from the target channel.

We made a slight modification to the gradient ascent algorithm. The objective function is the response of the center unit, rather than the entire channel. Using the top-10 patches as the initial images, we computed ten gradient ascent images, one for each patch. To isolate the contributions of gradient ascent from the original top patch, we subtracted the latter from each gradient ascent visualization. This left only the differences between the modified and original images, allowing us to more clearly see the features that were added through gradient ascent. Then, similar to the guided backpropagation method, we averaged the RGB channels then took the absolute value of the pixel values. This resulted in ten 2D gradient ascent visualizations. Finally, we averaged those visualizations to get a single 2D visualization for each unit. I have made an interactive website specifically for gradient ascent:
https://tonyfu97.github.io/DeepDreamAtNeuronLevel/index.html.

Advantages:

- As we noted in our discussion of the natural image method, one potential limitation is that there may not be an optimal natural image patch in our dataset that maximally

activates a given unit. To address this challenge, gradient ascent can be used to iteratively modify an image so that it becomes increasingly optimal with each iteration.

Disadvantages:

- Unlike guided backpropagation, which can be done in a single forward and backward pass, gradient ascent requires multiple steps to converge to a visualization. As a result, more work needs to be done to determine the appropriate number of steps required to obtain a useful visualization. Additionally, deeper layers typically require more steps, making the process slower than guided backpropagation.
- Similar to guided backpropagation, gradient ascent visualization results depend heavily on the initial image, which means that visualizations still rely on prior information in natural images. In other words, this method is not a true "ground truth" representation of the network's architecture and weights since it requires external stimuli.
- Gradient ascent aims to maximize local responses, but there is a risk of getting trapped in local optima. Although we experimented with the Adam optimizer, which incorporates a momentum term to improve convergence, the resulting visualizations were found to be excessively noisy.

Occlusion

Following the concept of Zhou et al. (2015), we overwrote the top and bottom images with occluders and measured the change in response compared to the original images. Occluder size and stride were scaled in proportion to the theoretical maximal RF size for units in the layer under study. The square occluder width was `w = 0.1M`, where `M` was the maximal RF width for the layer. The stride was: `s = 0.5w`. Both `w` and `s` are rounded down to the nearest integer.

For each occluder, the pixel in the discrepancy map corresponding to that at the center of the occluded location in the image was incremented by the absolute value of the difference between the response to the occluded and unoccluded image. We averaged the maps across the top 100 images to get the final discrepancy map. Each occluder was a different random noise patch, where each pixel was uniformly distributed over `[-1, +1)`, set independently for the R, G and B color planes and independently across space.

Advantages:

- Occluder discrepancy provides an alternative visualization method to the gradient-based approaches discussed earlier and can be used to validate the results obtained from these methods.

Disadvantages:

- Occluder discrepancy is computationally expensive as it requires one forward pass of the images for each occluder location, which can take a significant amount of time for large models. This can limit the scalability of this method.
- For units that are already driven by high-frequency features, substituting the original patch with a random patch, which is also high-frequency, may not lead to a significant difference in the response, making it difficult to measure the importance of those specific regions in the input image.

Comparing two maps: Mappability metrics

We define a unit as "mappable" when the neurophysiologically-inspired method produces a map that closely resembles the ground-truth map. To evaluate the similarity between maps, we use basic approaches: (1) calculating the Pearson correlation coefficient between the maps, (2) calculating the Intersection Over Union (IoU) of the RF regions, and (2) computing the distance between the centers of the RFs in the maps. The center can be determined in one of three ways: (i) center of mass (CoM), (ii) hotspot peak, and (iii) 2D Gaussian fit.

Pearson correlation

The Pearson correlation coefficient (i.e., r-value) is an intuitive metric for assessing the similarity between two number sequences. We first normalized the pixel values (to be from 0 to 1) and applied Gaussian smoothing to the flattened RF maps of both the neurophysiologically-inspired and ground-truth methods. These 1D arrays were Gaussian-smoothed using a standard deviation of $1/30\ M$, where M represents the MRF size. Empirical observations indicated that occlusion maps yielded optimal results when smoothed with an standard deviation between $1/30$ and $1/10\ M$, as this range effectively removed local texture while preserving the overall shape of the map (**Supplementary Figure 5**). The Pearson correlation coefficient was then computed using scipy.stats.pearsonr.

Despite its simplicity, the Pearson correlation coefficient has some limitations. The presence of many near-zero pixels around the boundary of most maps may introduce a bias toward positive correlation. **Figure 7** illustrates this phenomenon, with a concentration of pixel values at the lower left corners of the scatterplots. Intuitively, we would expect a negative r-value for the unit in the left panel (Conv3-50), but the presence of near-zero pixels results in a positive r-value instead. Eliminating pixels with values below 0.1 does not address this issue, as removing pixels from both images can lead to further misinterpretation of their similarity. A more sophisticated method is needed to provide a comprehensive comparison between maps.

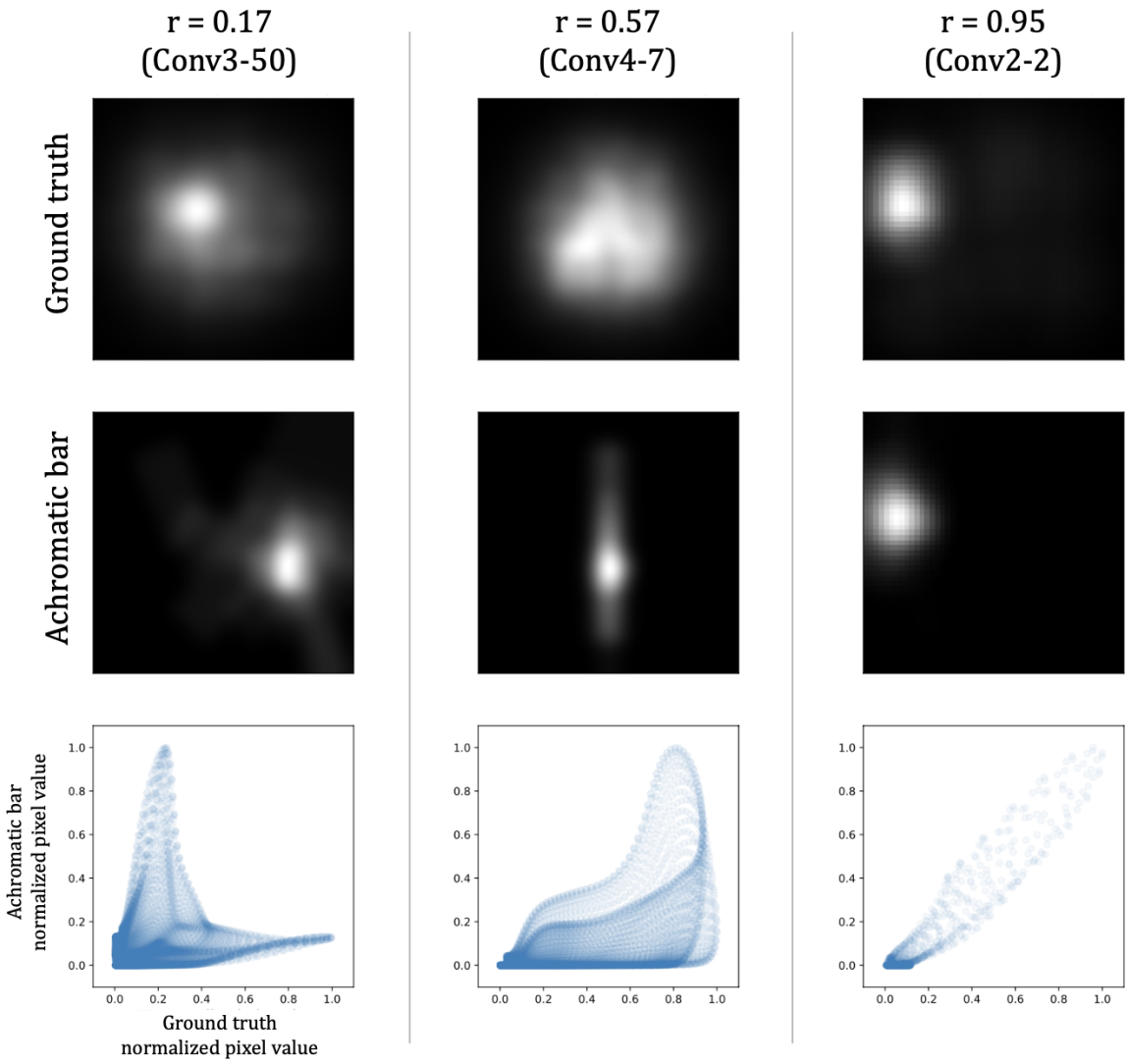

Figure 7. Three example units from AlexNet, assessed using Pearson correlation. Each column corresponds to a specific unit in AlexNet (e.g., Conv3-50 indicates unit no. 50 of Conv3). The top row displays ground truth maps, the middle row presents maps generated with achromatic bars, and the bottom row features scatter plots of normalized pixel values. From left to right, the three units/columns exhibit low, medium, and high correlation values, respectively. It is important to note that a majority of the maps appear black (with pixel values of zero). This observation is also evident in the scatter plots, where a high concentration of data points in the lower left corners results in a positive bias for correlation values. The scatterplots also appear artificially smooth, a result of Gaussian smoothing during the preprocessing stage.

Intersection Over Union (IoU)

Intersection over Union (IoU) is a metric commonly used to assess the similarity between two regions. Mathematically, it is defined as follows:

```
IoU = (Area of Intersection) / (Area of Union)
```

The pseudocode of the algorithm is provided below. We used a simple implementation of IoU: after Gaussian-smoothing and normalizing the two maps, we converted them to grayscale by taking the maximum value of each pixel across the RGB channels. The maps were then binarized using a threshold of 0.5. The intersection and union were computed using elementwise AND and OR operations, respectively. To achieve an IoU of 1, the two regions must not only overlap but also match perfectly. However, it is important to note that IoU does not quantify the distance between two non-overlapping regions.

Distance between centers

We can first summarize the ERF location using a few simple statistics and then compare the statistics of the two maps with each other. Here, we introduce three simple methods to define the center of the ERF given a RF map: center of mass, hot spot, and 2D Gaussian fit.

- **Center of mass (CoM):**
  For each map, we computed the center of mass as well as the radius that contains a 0.1, 0.5, and 0.9 of the mass.
- **Hotspot peak:**
  We find the coordinates of the maximum pixel.
- **2D Gaussian fit:**
  We fit 2D RF maps to Gaussians using SciPy's curve_fit() function. We determined the following parameters: `cx, cy` (the center coordinates), `s1, s2` (the standard deviations of the major and minor axes, respectively), `o` (the orientation of the major axis), and `a, b` (the amplitude and baseline offset, respectively). We measure the goodness of fit using the fraction of explained variance (fxvar), which has the formula:

  ```
  fxvar = 1 -  Var[map - fit] / Var[map]
  ```

  where `map` is the flatten array of the original RF map, and `fit` is the flatten array of the fit results.

For Hotspot peak and 2D Gaussian fit, we first smoothed the map with a 2D Gaussian filter (scipy.ndimage. gaussian_filter). The standard deviation of the smooth was set to 1/30 M (where M is the MRF size). For CoM, we only smoothed the RF maps generated by occluder mapping due to their sparse appearance. The algorithm is provided below:

The Gaussian center is only used when maps pass a goodness of fit criterion (`fxvar > 0.7`), whereas the center of mass and hotspot methods are valid for all maps. However, in CNNs, we often observe RFs that are divided into distinct islands. This phenomenon may arise from the mapping methods employed, which generally isolate the excitatory regions in the 'top' maps and the inhibitory regions in the 'bottom' maps. This fragmentation complicates the process of identifying the centers. As demonstrated in the right panel of **Figure 8**, when RF maps

comprise separate islands or subregions, the CoM is easily influenced, tending to find a center point even in the absence of mass. Conversely, the Gaussian fit and hotspot peak methods usually select one of the islands to represent the RF. Determining the correctness of these methods is subjective, as it largely depends on the definition of the RF.

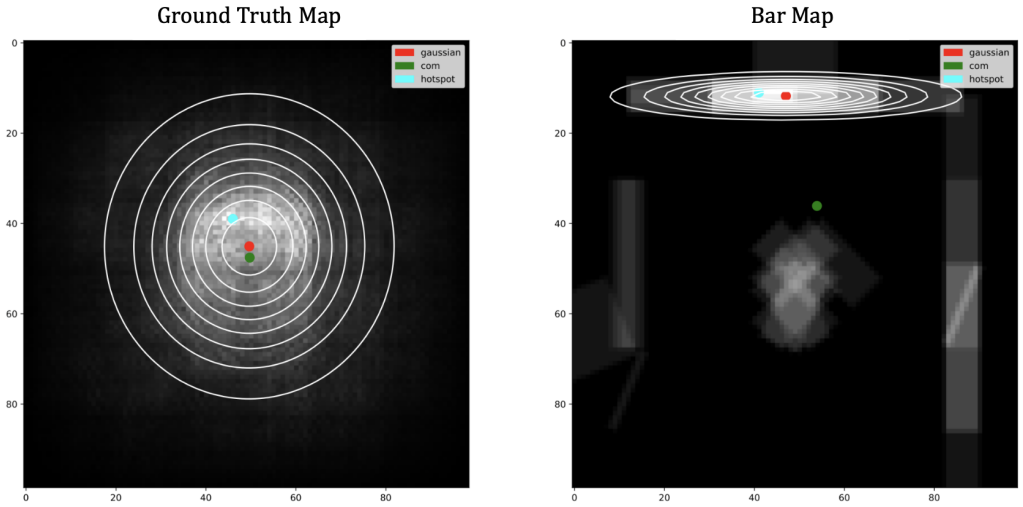

Figure 8. (**Left**) The RF map obtained using the ground-truth method. (**Right**) The RF map generated using achromatic bars. Both maps are fit using the three center-finding methods. The colored dots denote the RF center as defined by each method: 2D Gaussian (red), CoM (green), and Hotspot peak (cyan). The unit is AlexNet Conv3-7.

Discrepancy analysis: Diagnostic metrics

We expect to observe a wide range of mappability among the units, as measured by mappability metrics introduced previously. To explain why some units are more bar-mappable, we have formulated several hypotheses:

1. Ground-truth map is not correct.
2. Our bar stimulus set does not contain features that will drive the units effectively.
3. The low performance of achromatic bars is due to the units being color selective.

The first hypothesis can be tested by comparing the map generated by different types of ground-truth methods. To test the latter two hypotheses, we introduce some diagnostic metrics that will be applied to the units.

Fraction of top-10 natural image response average (F_{nat})

To evaluate our second hypothesis, which states that certain units are not effectively mappable with bars due to insufficient driving by the bars, we devised a metric called F_{nat}, an abbreviation for Fraction of NATural image response average. This metric compares the maximum response elicited by a bar to the average maximum responses of the top 10 natural images. Fnat is calculated by dividing the response of the top-performing bar (i.e., the bar that

produces the strongest response in the unit) by the average responses of the top 10 natural images. The formula for Fnat is as follows:

```
Fnat = (top1 bar response) / (top-10 natural image response avg)
```

We expect bar-mappable units to be high F_{nat}, values.

Color rotation index (CRI)

One way to quantify the color-sensitivity of a unit is to show it some images repeatedly, have their RGB channels rotated every time, and then calculate the response variability. An RGB channel can be permuted in 6 ways, giving us 5 permutations in addition to the original image. A unit is said to be color-sensitive if the rotation leads to a big change in its response. Using a random set of 1000 images (out of 50,000 natural images, with replacement), we compute the color rotation index (CRI), for each unit. The formula is as follows:

```
CRI = E[SD_cr] / SD_images
```

where `SD_cr` is the standard deviation of the responses to the six different color rotations, and `SD_image` is the standard deviation of the responses to the original 1000 images.

Web app (RF Playground)

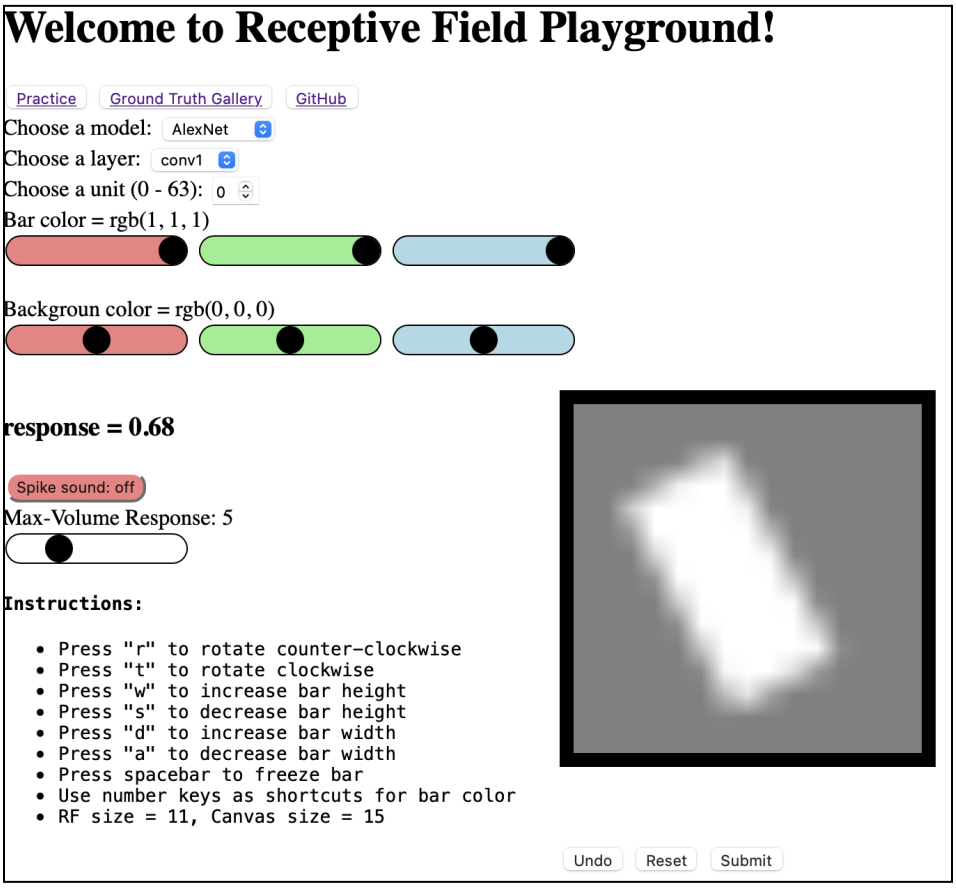

Figure 9. User Interface of Receptive Field Playground Web App. At the top of the web page, a global navigation bar provides access to three other pages: a practice page where users can define their own ground-truth map and practice RF mapping skills, a gallery page where users can browse ground-truth guided-backpropagation images, and the GitHub repository for the website. Below the global navigation, there is a menu for selecting the model, layer, and unit, as well as sliders for choosing the colors of the bar and background. On the right, a canvas allows users to hover their mouse to control the position of a bar stimulus. The unit's response will be updated on the left side of the web page. The "Spike sound" button can be toggled to play spikes whenever the response is positive, with louder spikes indicating a greater response. The "Max-Volume Response" slider adjusts the response value at which the spike volume saturates to the maximum volume. Additional instructions are displayed on the left-hand side.

Web app link: https://tonyfu97.github.io/rf_playground/

One comment received during the journal club presentation about our project was that a trained electrophysiologist might perform better than automated RF mapping due to their experience and skills. Therefore, to enable trained electrophysiologists to map the RF of CNN units, we created a web application that allows anyone to map the RF of AlexNet, Vgg16, and ResNet18 using bar stimuli on their browser in real-time. The user interface consists of a menu that allows users to select a unit and sliders to control the color of the bar and the background. When the user hovers their mouse over the canvas, a bar moves with the mouse. The image is fed

into the selected CNN, and the unit's response is displayed. If the user toggles the unmute button, they can hear simulated spike sounds, with louder sounds indicating a more positive response (**Figure 9**).

To map the RF, users can click on the canvas to draw the vertices of a polygon that will be used as the boundary of the ERF. Each time the user clicks on the canvas, a line connects the current point with the previous point to form an edge. Users can press Enter to close the polygon, which is then filled with color to indicate the ERF region. After clicking the "Submit" button below the canvas, the ground-truth guided backpropagation map is fetched from AWS storage. The browser then calculates two mappability metrics (Pearson correlation coefficient and Intersection Over Union) and displays them, along with the ground truth map (**Figure 10**).

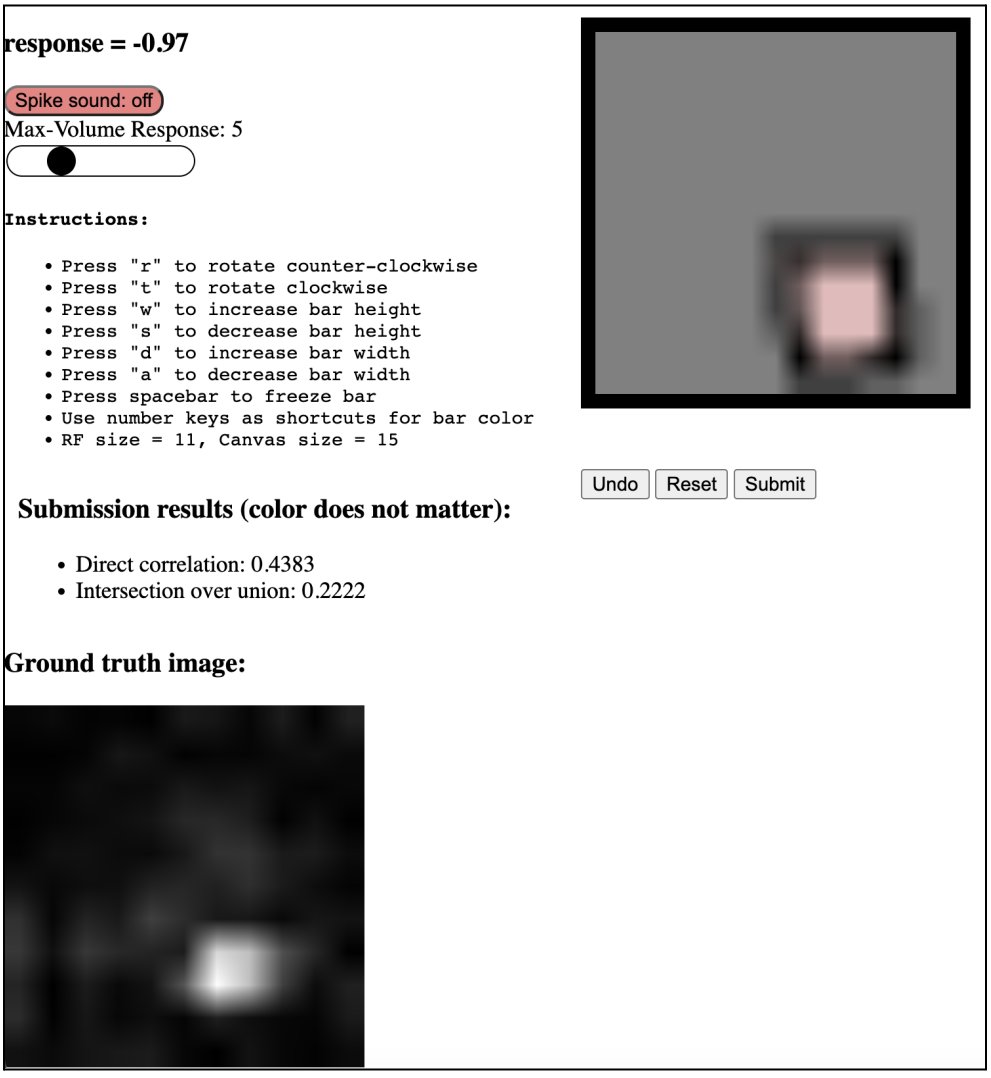

Figure 10. Example use of the RF Playground Web App. Users can draw the ERF boundary by clicking on the canvas to indicate the corners of the polygon, and then pressing enter to close the shape. After clicking "Submit", the drawn polygon is compared with the ground-truth guided backpropagation (which is displayed after the “Submit” button is clicked) using Pearson Correlation and Intersection over Union (IoU).

The website was inspired by Elliot Waite's real-time handwritten digit recognition web app (https://github.com/elliotwaite/pytorch-to-javascript-with-onnx-js). Under the hood, the pre-converted, truncated pre-trained CNNs in Open Neural Network Exchange (ONNX) format run in the browser using the ONNX.js library. The web page actually draws three canvas images simultaneously, with two canvases hidden from view. The visible canvas is where users interact, while the other two canvases are separated for specific purposes: one canvas contains only the bar and is fed to the CNN, and the other contains only the polygon and is used to compute the mappability metrics. The current web app is crude, and we plan to enhance its design and make it more visually appealing in the near future.

Results

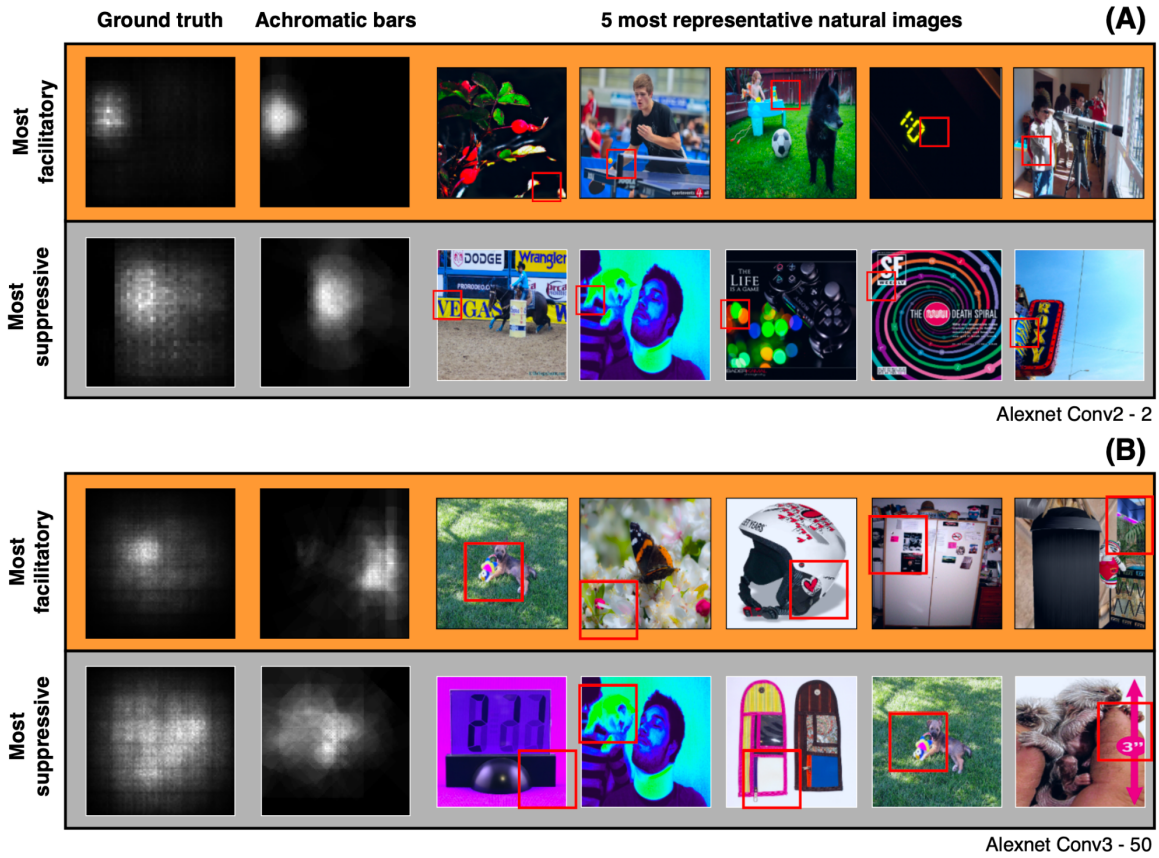

Figure 11. Examining good (top) and poor (bottom) bar-mappability using representative examples. (**A**) AlexNet Conv2-2 unit is highly responsive to objects located in the left mid-upper corner of its RF while being suppressed by objects in the center of the MRF. This is evident in both the ground truth map and the achromatic bar map, making the unit bar-mappable. The top five representative natural images also show the most facilitatory image patches (boxed in red) contain a bright spot on the left. Interestingly, the five most suppressive image patches are all located on the left edge, indicating that the unit's response is most suppressed when there is nothing on the left and bright spots are present in the middle of the MRF. (**B**) AlexNet Conv3-50 unit responds strongly to a magenta spot roughly at the center of the RF, as shown in both the ground truth map and the top five most facilitatory natural images. However, the achromatic bar map shows a contrasting pattern, suggesting the ERF is on the right of the MRF. This unit is therefore not bar-mappable by visual inspection. This discrepancy may arise from the achromatic nature of the bars and the unit's strong color selectivity towards magenta.

Diminishing mappability by achromatic bars

We begin by comparing the accuracy of the achromatic bar mapping paradigm against our ground-truth method in terms of its ability to find the center of ERF. **Figure 11 A** shows an example of a bar-mappable unit from Alexnet Conv2. The unit responds strongly whenever a bright spot appears on the left the image patch (orange row). The same unit is suppressed when the left side is empty and the rest of the image patch is colorful (gray row). Both the ground-truth

(guided backpropagation) maps and the achromatic bar maps correctly capture the locations and sizes of the most facilitatory and the most suppressive fields, indicating the reliability of our ground-truth method and the unit's mappability using black and white bars. However, numerous units are not bar-mappable. **Figure 11 B** shows a unit from Alexnet Conv3, and its achromatic bar map incorrectly places the most facilitatory field (orange row) at the side that is opposite to the ground-truth.

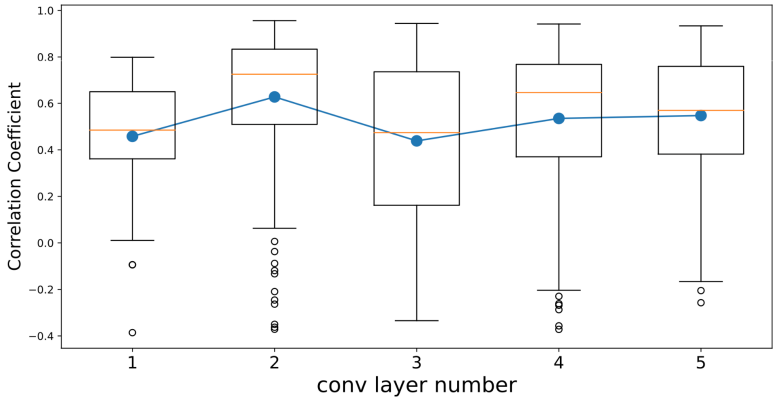

Figure 12. Distribution of Pearson correlation coefficients between the (facilitatory) ground-truth (guided backpropagation) maps and achromatic bar maps across different layers in AlexNet. No discernible trend is observed. However, caution is advised in interpreting these results, as Pearson correlation coefficients may overestimate the similarity when the maps are sparse, containing a significant number of near-zero pixel values.

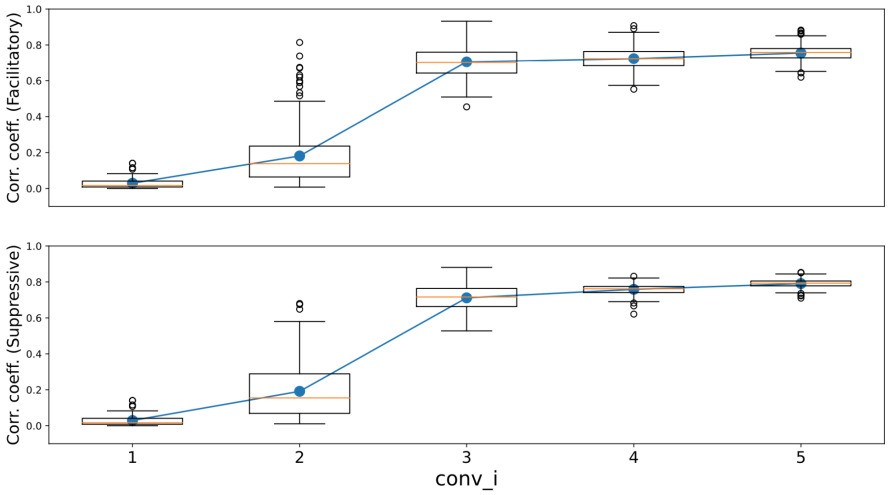

Figure 13. Sparsity of ground-truth (guided backpropagation) maps of AlexNet, where sparsity is defined as the proportion of pixels with values below an absolute tolerance of 0.01. A sharp increase in sparsity is observed at Conv3 for both facilitatory and suppressive ground-truth maps, indicating that the ERF is concentrated in specific regions of the MRF.

In the methods section, we introduced a simple approach to quantify a unit's bar-mappability by calculating the Pearson correlation coefficient between the ground-truth (guided backpropagation) map and the bar map. **Figure 12** shows the distribution of Pearson correlation coefficients in Conv1 to Conv5 of AlexNet. No clear trend was observed as we

moved to deeper layers. However, as previously discussed, Pearson correlation coefficients may overestimate the similarity between the maps, particularly when the map is sparse (i.e., having many near-zero pixels, in this case using an absolute tolerance of 0.01). **Figure 13** shows a sharp increase in sparsity in Conv3 of AlexNet, which might lead to an overestimation of the similarity between the ground-truth and bar maps. A similar trend was observed in Vgg16 (**Supplementary Figure 6**).

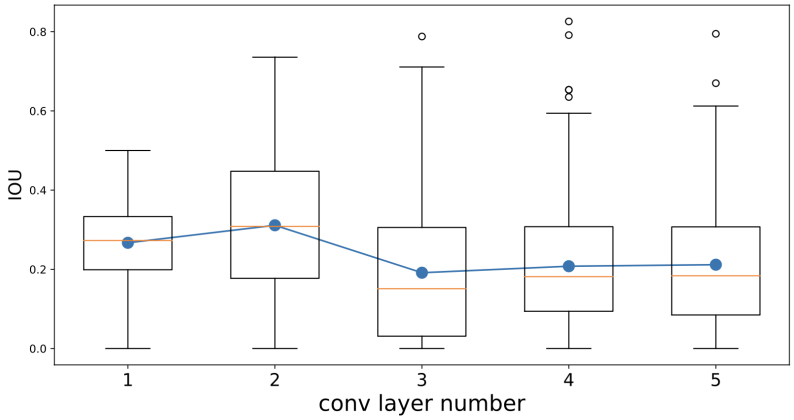

Figure 14. Distribution of Intersection over union (IoU) between the (facilitatory) ground-truth (guided backpropagation) maps and achromatic bar maps across different layers in AlexNet. The IoU peaks at Conv2 and deteriorates afterwards.

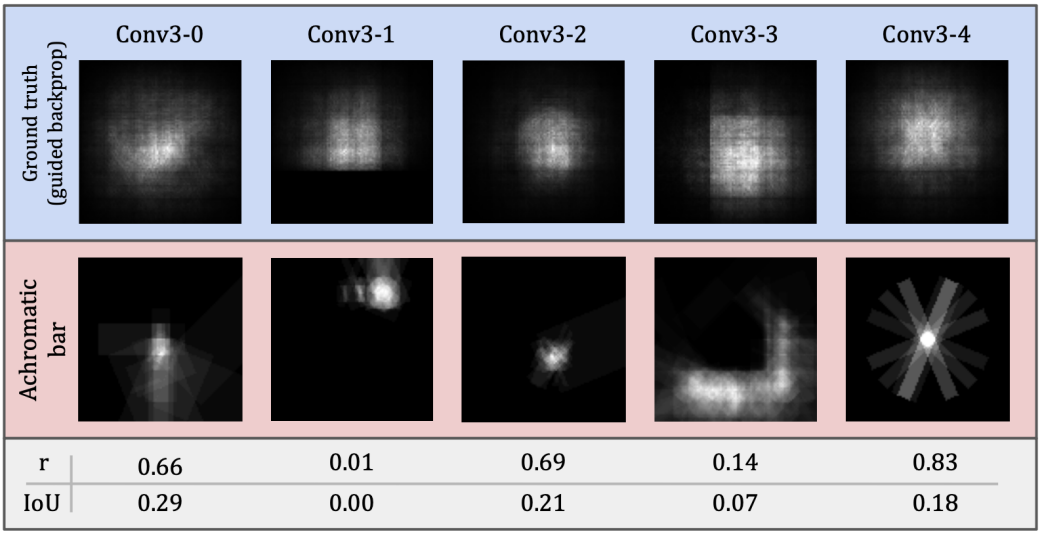

Figure 15. Receptive field maps of the first five units of AlexNet Conv3. The top row (blue) displays the ground-truth (guided backpropagation) maps, where the ERFs are typically 2D Gaussians centered within the MRF. The middle row (red) presents the achromatic bar maps, which are usually smaller in size and more irregular in shape. The bottom row (gray) indicates the respective correlation coefficients (r) and Intersection over Union (IoU) values for the two maps. The correlation coefficient (r) tends to overpredict similarity, while the IoU values provide a more accurate representation.

An alternative metric to the Pearson correlation coefficient is Intersection over Union (IoU). This metric addresses the issue of an abundance of near-zero pixels by considering only

the non-zero regions in the map, thus quantifying the degree of overlap between the ERFs mapped by the ground-truth and bar maps. **Figure 14** displays a similar trend as **Figure 12**; however, note the decrease in IoU in the latter layers. This is more consistent with our qualitative observations. **Figure 15** showcases the first five units of the AlexNet Conv3 layer, where the achromatic bar map struggles to accurately locate the ERF. The correlation coefficient overestimates the similarity, while the IoU values align with our expectations.

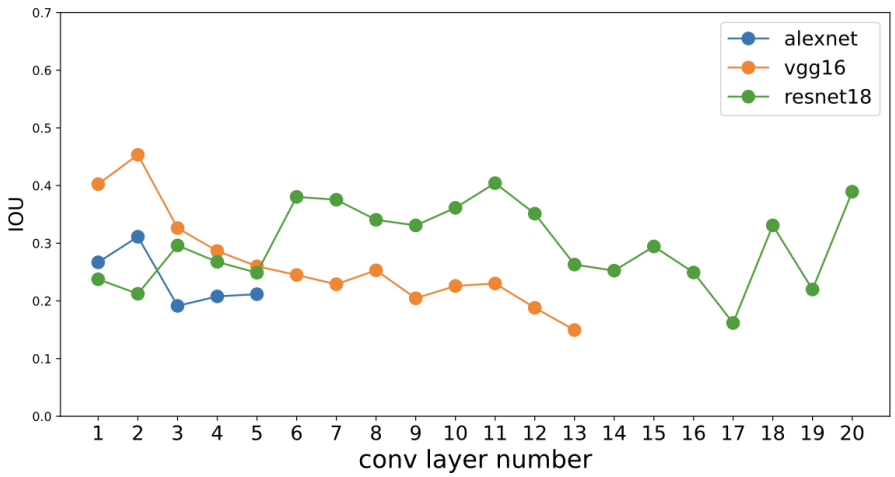

Figure 16. Average intersection over union (IoU) trend between the (facilitatory) ground-truth (guided backpropagation) maps and achromatic bar maps across different layers in the models. In AlexNet and Vgg16, the average IoU decreases as the layers progress. In contrast, there is no discernible trend in ResNet18.

A similar trend is observed in Vgg16 (**Figure 16**), where the IoU decreases over layers, suggesting a decrease in bar-mappability. However, an interesting trend is observed in ResNet18, where the peak in IoU occurs much later: instead of peaking in Conv2, as in AlexNet and Vgg16, the IoU of ResNet18 peaks at Conv6. It is worth noting that ResNet18 has special residual connections, which may contribute to the unique trend observed here. Additionally, for ResNet18, Conv16-20 are larger than the input image itself (**Supplementary Figure 7**). Furthermore, Conv8, Conv13, and Conv18 are convolutional layers on skip connections, so caution should be exercised when interpreting these layers.

However, IoU does not penalize the distance between two non-overlapping ERFs or differences in sizes. In other words, low IoU values could be due to a large offset in center location, significant size differences, or both. To focus solely on comparing the center locations of the ERFs without considering size differences, a distance metric can be used. For instance, we can fit a 2D Gaussian to each map and then compare the center coordinates of the two maps.

Figure 17 presents a set of scatter plots comparing the x-coordinates of the 2D Gaussian-fitted ERFs for the ground truth (guided backpropagation) and achromatic bar maps. In Conv2 (left-most scatterplots), the x-coordinates of the two maps match for most units, resulting in most data points lying on the y=x diagonal (with r = 0.81 for the facilitatory map, and r = 0.56 for the suppressive map). However, as we move into deeper layers, the scatter plots become elongated. The ground-truth x-coordinate remains around the center, while those of the bar maps

tend to deviate, reflecting our qualitative observations (**Figure 15**). The y-coordinate shows a very similar trend, as summarized in **Figure 18**.

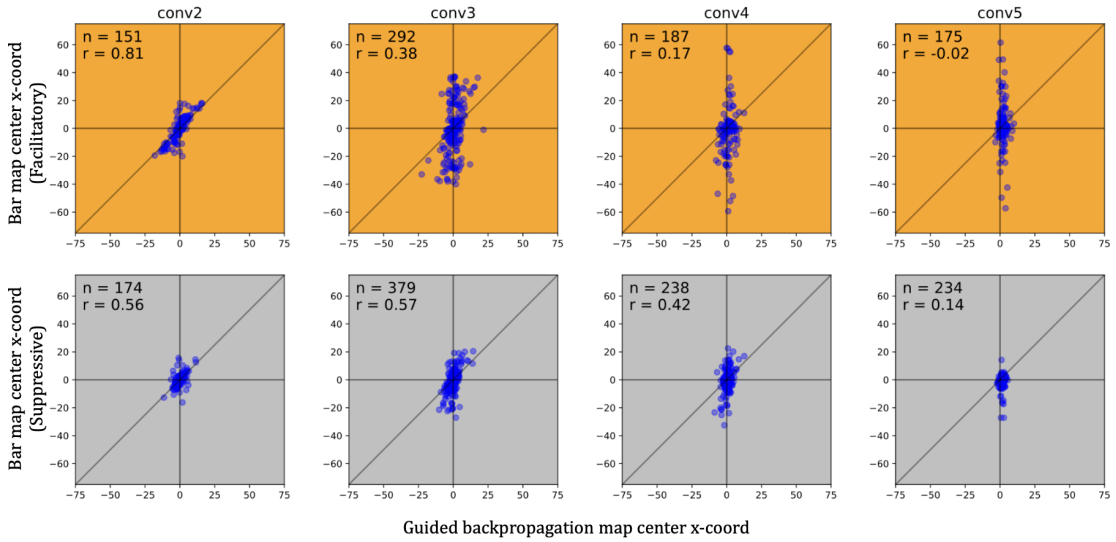

Figure 17. Comparison of x-coordinates of the ERF for achromatic bar maps and ground-truth (guided backpropagation) maps obtained from 2D Gaussian fit in AlexNet. The horizontal axis represents the ground-truth ERF x-coordinates, and the vertical axis represents the achromatic bar map ERF x-coordinates. There is a strong correlation in Conv2, but it quickly diminishes in deeper layers. The scatterplots also reveal considerable variation in the x-coordinate of the bar map ERF, while the ground-truth ERF is more frequently centered, resulting in vertically elongated scatterplot clusters. Note that some units were excluded from the analysis because they did not have a good fit to a 2D Gaussian for at least one of the two maps (good fit is defined as having a Fxvar over 0.7). As a result, for example, the number of units analyzed in the Conv2 facilitatory map is 151 instead of 192.

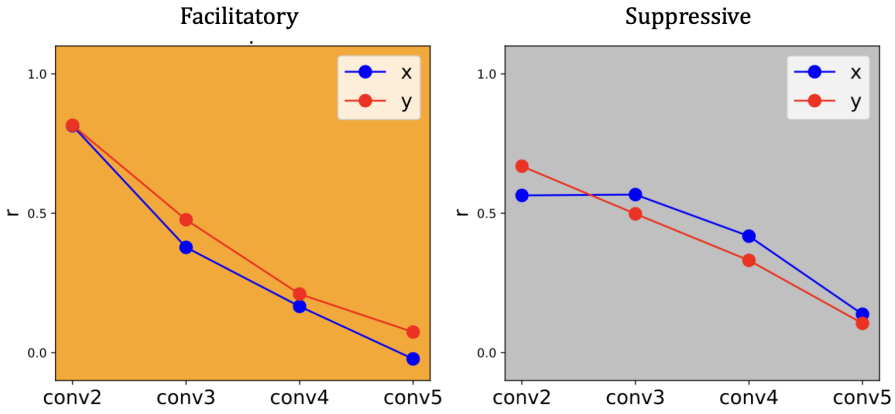

Figure 18. Trend of bar mappability in AlexNet as measured by the correlation between x- and y-coordinates of ground-truth and achromatic bar map ERFs. For both facilitatory and suppressive maps, the correlation of the coordinates decreases across layers, suggesting a reduction in mappability by achromatic bars.

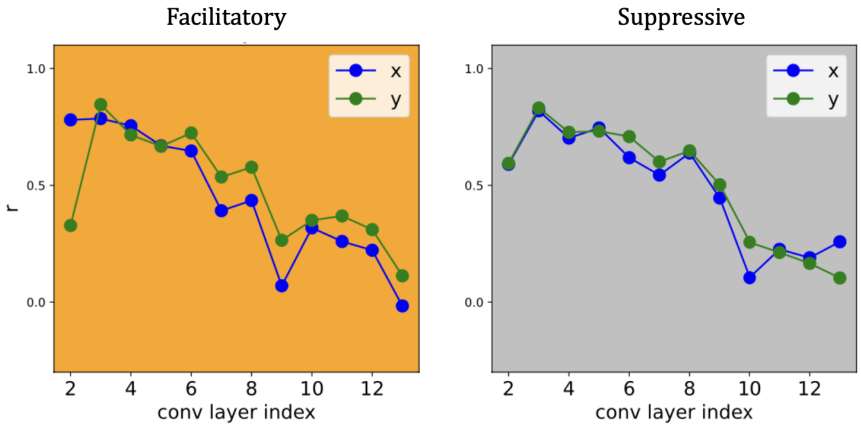

Figure 19. Similar to **Figure 18**, but for Vgg16.

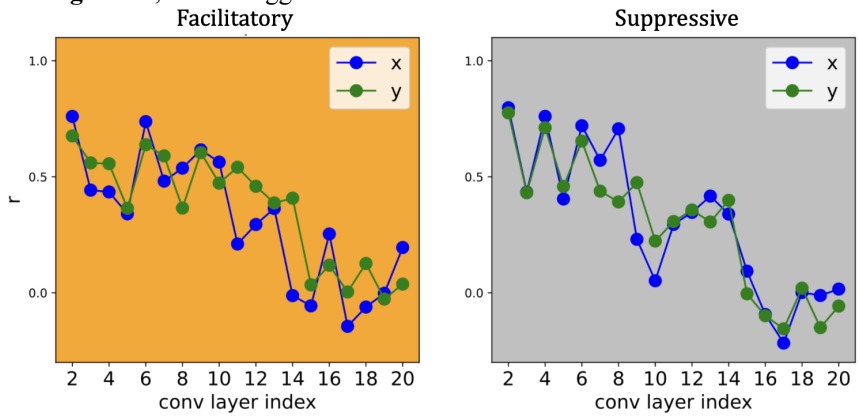

Figure 20. Similar to **Figure 18**, but for ResNet18. The trend of correlation coefficient (r) is also more irregular than in AlexNet and Vgg16, exhibiting a zig-zag pattern across layers.

A similar pattern is observed in Vgg16 (**Figure 19**), where the correlation between the x- and y- coordinates decreases across layers. However, in ResNet18, a zigzag pattern is observed (**Figure 20**), suggesting that mappability tends to fluctuate. This pattern may be introduced by the presence of skip connections. We will discuss ResNet18 further in the discussion section.

Diminishing mappability by chromatic bars

Upon using achromatic bars to produce the bar maps, we quickly noticed that many color-selective units had inaccurate bar maps. This prompted us to create a new set of bar stimuli with colors. The challenge was to make the background of the bar as neutral as possible, ensuring that most of the response could be attributed to the bar and not the background. Consequently, we used a background of zeros, effectively turning off the Conv1 weights at those pixel locations. In many cases, some units that were previously unmappable by achromatic bars became mappable. For example, Conv3-50 (introduced in **Figure 11 B**) is now mappable using predominantly magenta bars (**Figure 21**).

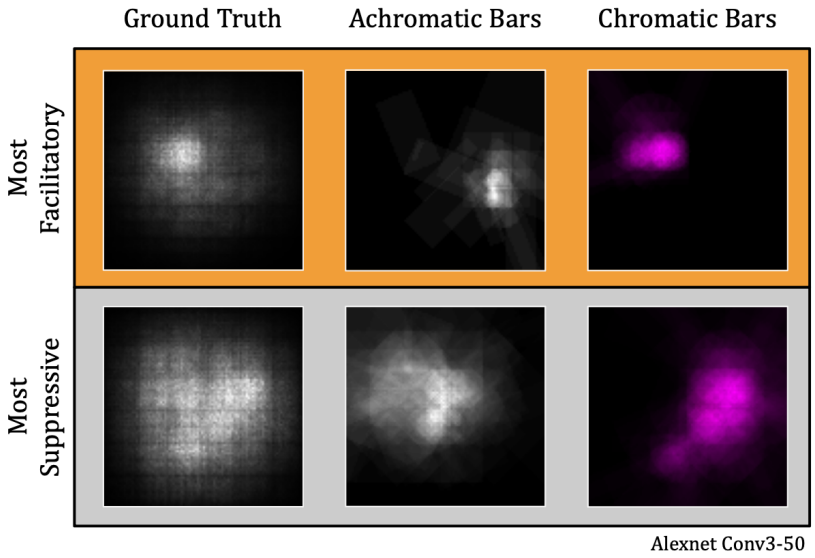

Figure 21. Chromatic bars effectively map AlexNet Conv3-50, which was previously unmappable with achromatic bars (middle column). The chromatic bar maps (right column) accurately highlight the region where the neuron can be excited (top row) and suppressed (bottom row), although they may underestimate the ERF size.

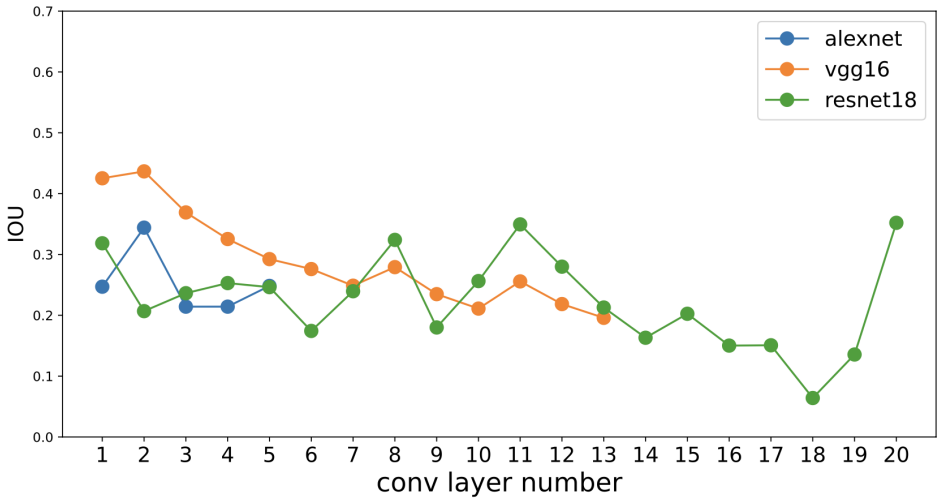

Figure 22. Average intersection over union (IoU) trend between the facilitatory ground-truth (guided backpropagation) maps and chromatic bar maps across different layers in the models. The results are qualitatively similar to **Figure 16** for AlexNet and Vgg16. Once again, a zip-zag trend is observed in ResNet18.

However, while the use of color bars improves the mappability of certain units, it does not fundamentally alter the overarching trend of decreasing mappability. The patterns evidenced in **Figure 22** align closely with those in **Figure 16**, indicating that, on average, the overlap between most bar maps and the ground truth remains limited, particularly in the deeper layers. This similarity extends to the coordinates of the 2D Gaussian fit of the maps, where a persistent decrease in correlation between the coordinates of the two maps is observed. **Figure 23** confirms these trends, showing a pattern analogous to that observed with achromatic bars.

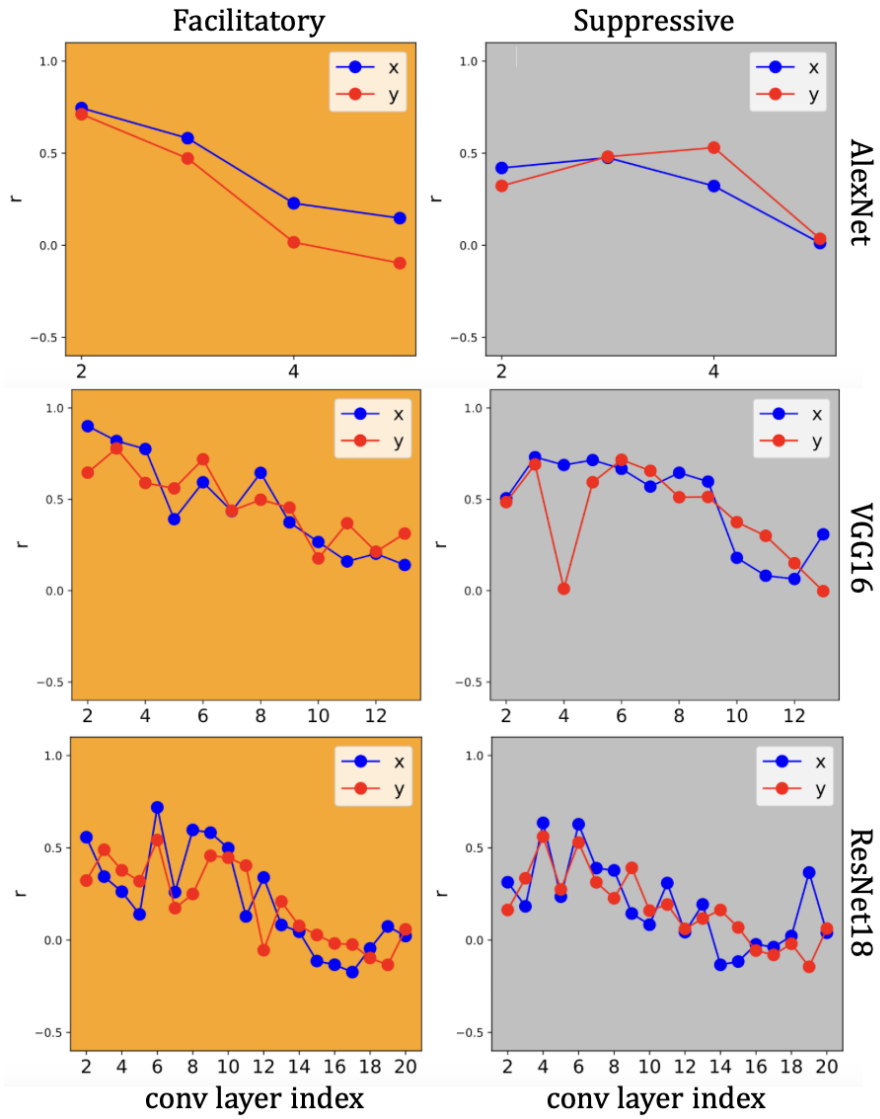

Figure 23. Trends in bar mappability for AlexNet, VGG16, and ResNet18. Mappability is measured by the correlation between x- and y-coordinates of ground-truth and color bar map ERFs. These trends mirror those observed in Figure alexnet_gt_rfmp4a_gaussian_fit_summary, **Figure 19** and **Figure 20**. Nonetheless, a distinct dip in the correlation of y-coordinates is observed in Conv4 of VGG16.

Testing hypothesis 1: Are ground truth maps reliable?

Upon observing our results, we began to question the reliability of our guided backpropagation ground truth maps. In the case of AlexNet, for example, the majority of units in Conv3 to Conv5 possess ERFs that follow a 2D Gaussian distribution (**Figure 24**). While it makes sense that ERFs are centered due to the higher amount of input received from preceding layers as opposed to peripheral ones, the prevalence of such patterns in Conv3 to Conv5 is surprising. Unlike their counterparts in Conv1 and Conv2, these ERFs do not exhibit distinctive shapes. Consequently, we sought an alternative ground-truth method to verify the results obtained from guided backpropagation. Inspired by the approach of Zhou et al. (2015), we

introduced occluders to the top and bottom images and measured the changes in response as compared to the original images. This method was chosen specifically due to its non-gradient-based nature, contrasting with guided backpropagation.

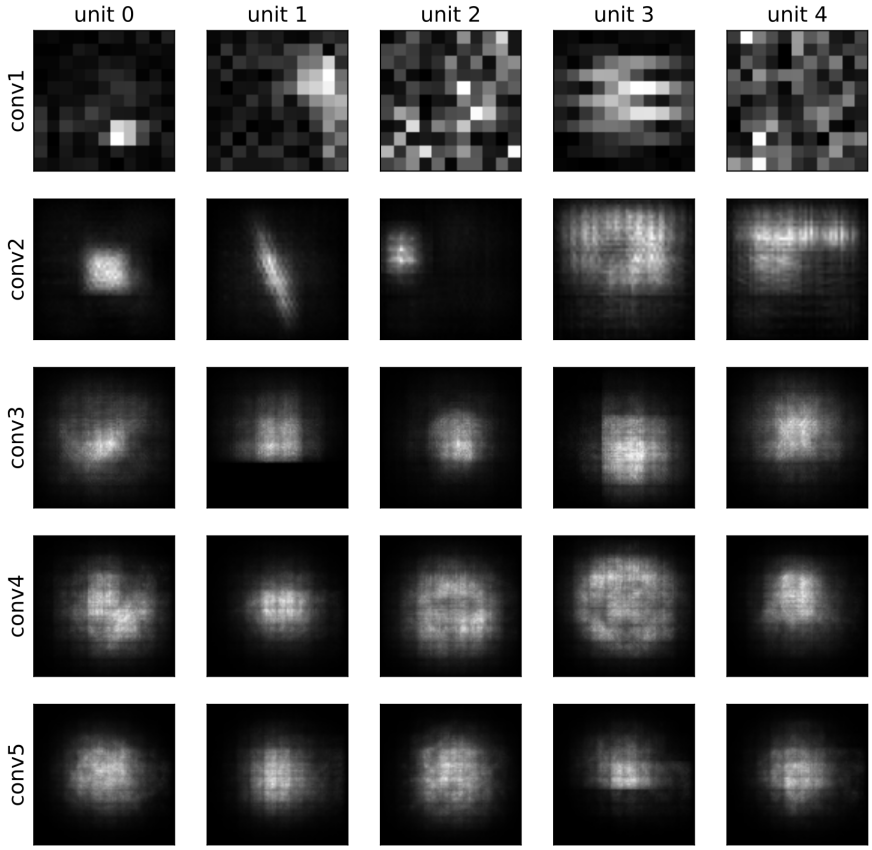

Figure 24. First five units of each convolutional layer within AlexNet. Compared to Conv1 and Conv2, the ERFs (depicted as white-ish pixels) in Conv3, 4, and 5 are noticeably more centered.

One of the initial observations we made about these occluder "discrepancy" maps is that their ERFs do not differentiate between facilitatory and suppressive regions, unlike the guided backpropagation (**Figure 25**). This is explainable as the discrepancy maps are derived by calculating the absolute value of the response difference, thereby recording both increases and decreases in the response. The absolute value is necessary to account for situations where a response decrease could either be due to the occlusion of a significant feature in the original image, or due to the introduction of high-frequency noise in an area where the unit prefers a blank space.

In order to compare between these discrepancy maps and the guided backpropagation maps, we first had to combine the facilitatory and suppressive maps to produce a composite map. This was particularly noticeable in the examples shown in **Figure 25**. Next, we calculated the correlation coefficient between the discrepancy maps and these composite maps. As depicted in **Figure 26**, there was almost perfect agreement between the two maps, lending credibility to our guided backpropagation approach.

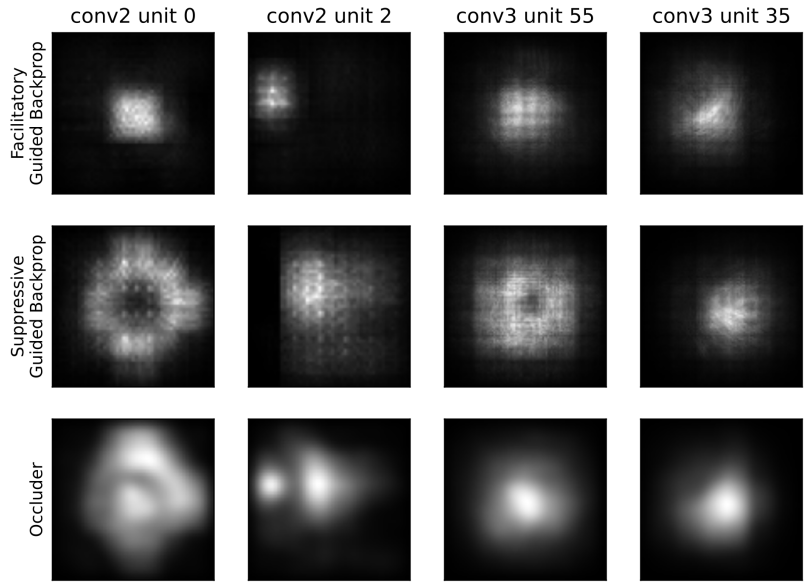

Figure 25. Some selected examples of guided backpropagation and occluder "discrepancy" maps. The top and middle rows represent the facilitatory and suppressive guided backpropagation maps, respectively. The bottom row displays the occluder discrepancy maps after they have been smoothed by a Guasian of standard deviation of 1/30 MRF. Upon visual inspection, it becomes evident that the relationship between the two types of maps essentially follows the formula: top + middle = bottom row.

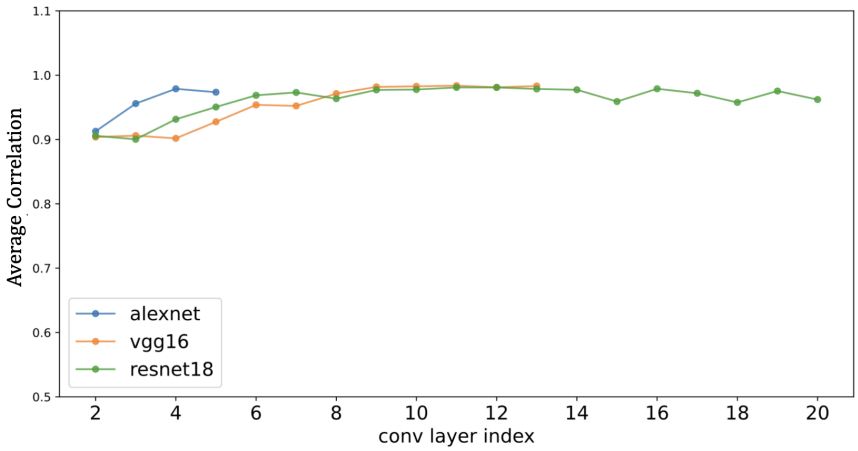

Figure 26. Correlation between two ground-truth maps - guided backpropagation and occlusion. It is important to note that occlusion is sensitive to both facilitatory and suppressive regions; therefore, it is compared with the sum of the facilitatory and suppressive maps of guided backpropagation. The two methods generate almost identical maps, as measured by the Pearson correlation coefficient.

Testing hypothesis 2: Do units not respond to bars as much as natural images?

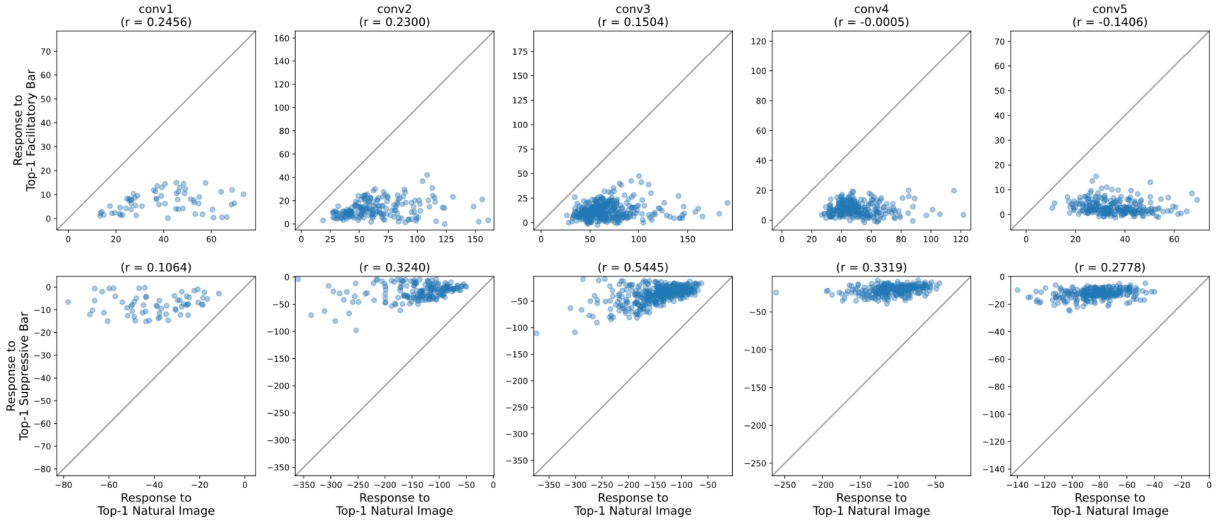

Figure 27. Comparing the responses of AlexNet's units to the top-1 natural images with their responses to the top-1 achromatic bars. The horizontal coordinates of each data point represent a unit's response to the top-1 most facilitatory (top row) or suppressive (bottom row) natural image patch, while the vertical coordinates indicate the response to the top-1 most facilitatory (top row) or suppressive (bottom row) achromatic bar stimulus. If a data point lies on the diagonal y=x line, this implies that the unit's response to the natural image patch and the bar are equal. In the facilitatory maps (top row), all units lie in the lower right triangle, signifying that their responses to bars are consistently weaker (less positive) than their responses to the natural images. In the suppressive map (bottom row), all units lie in the upper right triangle, indicating that their responses to bars are always weaker (less negative) than their responses to the natural images.

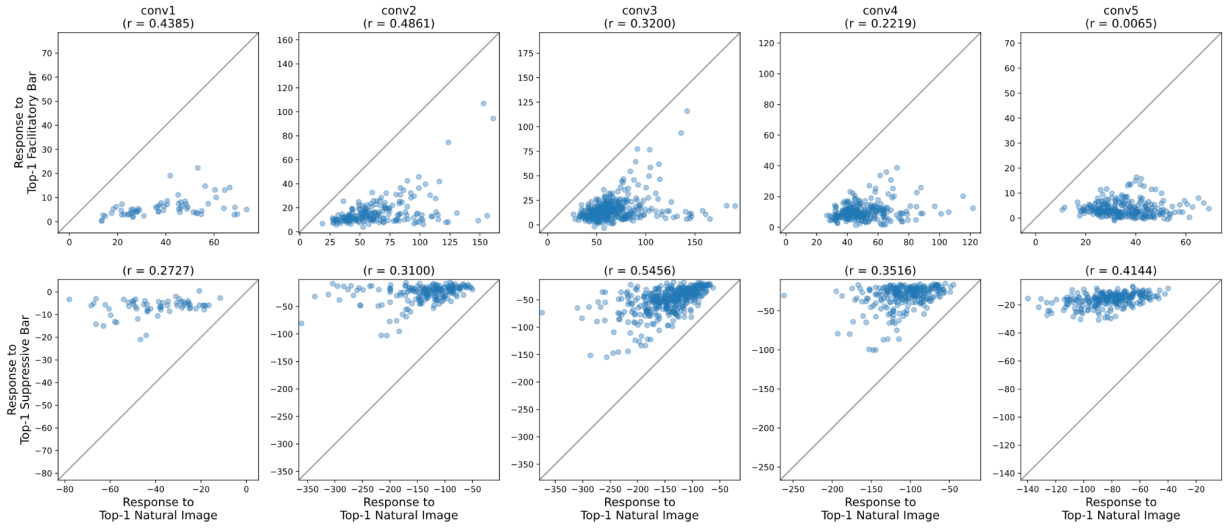

Figure 28. This figure is similar to **Figure 27**, but instead uses color bars. The introduction of color has amplified the response strength of certain units, as evidenced by their closer proximity to the y=x diagonal line compared to their positions with achromatic bars. However, none of these responses have managed to match the intensity of the response to the top-1 natural image patch.

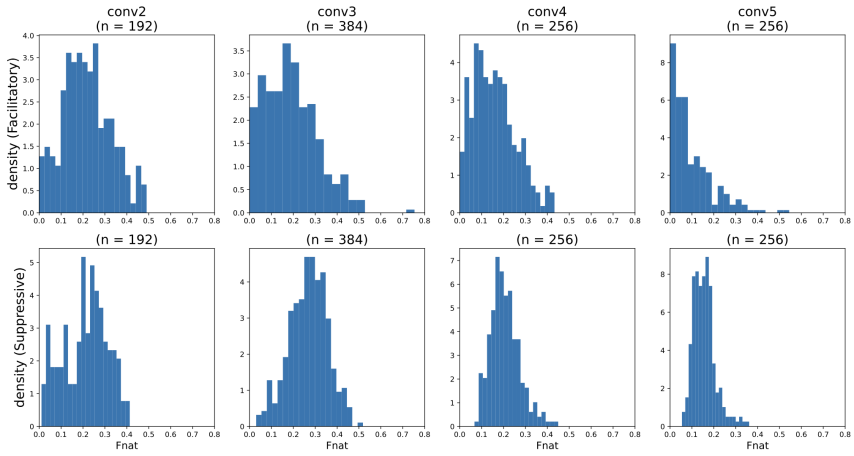

Figure 29. Distribution of Fraction of top-10 natural image response average (F_{nat}) statistics in AlexNet when using achromatic bars. The ratio of the response to the top-1 bar to the average response of the top-10 image patches consistently falls below 0.5. This implies that the most effective bar stimulus can't elicit even half the average response compared to natural images.

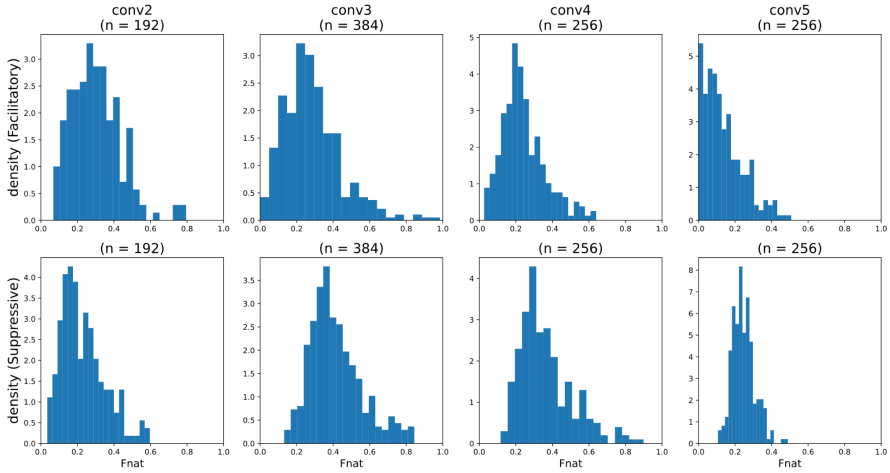

Figure 30. This figure is similar to **Figure 29**, but uses color bars.

We began to question whether simple stimuli like bars could ever elicit a response comparable to that produced by natural images. Our results soon confirmed our suspicions: many units generated blank bar maps, meaning that no bar stimuli produced a positive response in these units. To visualize this phenomenon, we plotted the unit's response to the most facilitatory and suppressive natural images and the responses to the most facilitatory and suppressive bars (see **Figures 27** and **Figure 28**). Interestingly, none of the units responded more strongly to bar stimuli compared to natural images. This held true for both achromatic and color bars.

To quantify the response mismatch, we devised a metric called F_{nat}, which stands for 'Fraction of NATural image response average'. The formula for this metric is provided in the Methods section. The distribution of F_{nat} values for achromatic bars mostly fell below 0.5 (**Figure 29**), indicating that most bars elicited responses less than half as strong as those to the natural image patches. While the introduction of color bars improved the responses of a small number of units, still none elicited a stronger response than natural images (**Figure 30**). This is

also apparent in **Figure 28**, where a small number of units are closer to the y=x diagonal compared to those in **Figure 27**.

Testing hypothesis 3: Is color sensitivity related to mappability?

Looking at the results from our second hypothesis test, we can see that adding color to the bar stimuli does help some units respond better, as shown in **Figure 21**. However, this improvement isn't seen in all units. **Figure 31** highlights this variation. In every layer of AlexNet, there are units that respond much more to color bars than to achromatic bars, but there are also units that respond similarly to both.

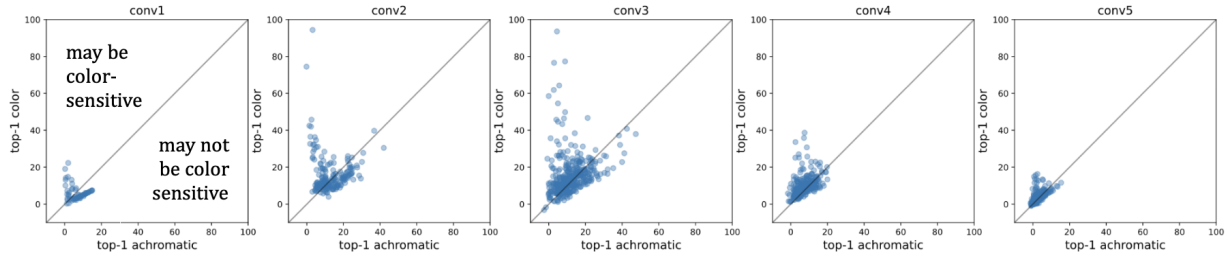

Figure 31. Comparing the responses of AlexNet units to the top-1 achromatic bar (horizontal axis) and the top-1 color bar (vertical axis). Across all layers of AlexNet, units can be distinctly categorized into two groups, separated by the diagonal y=x line. The group in the upper triangle demonstrates a substantially higher response to the top-1 color bar compared to the top-1 achromatic bar, suggesting these units may be color-sensitive. Meanwhile, units in the lower right triangle show comparable responses to both color and achromatic bars. Notably, this latter group does not align precisely with the diagonal but sits slightly below it, which may be attributed to the use of a gray background in the color bars

To quantify the sensitivity of units to colors, we devised the Color Rotation Index (CRI). This index measures the variation in a unit's response to 1000 randomly selected natural images when the RGB channels of these images are permuted in six ways. As illustrated in **Figure 32**, units with low CRI usually prefer grayscale patterns, whereas units with high CRI are more likely to favor specific colors, regardless of the patterns. The moderately positive correlation between CRI and the difference in responses to color and achromatic bars can be observed in **Figure 33.** However, **Figure 34** indicates that CRI, on its own, doesn't have substantial predictive power when it comes to the mappability of bars.

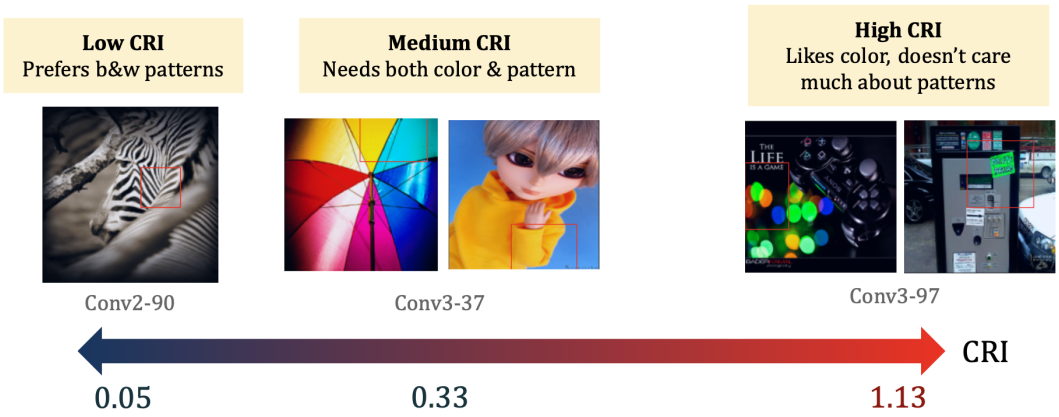

Figure 32. Example units with low, medium, and high Color Rotation Index (CRI) values. Units with low CRI generally exhibit a strong preference for particular (often grayscale) patterns, whereas those with high CRI typically favor specific colors, regardless of the shape. Units with a medium CRI demonstrate a balanced preference for both color and patterns.

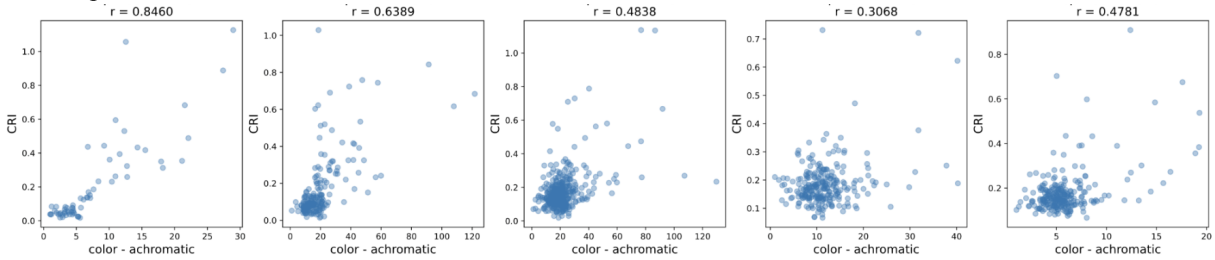

Figure 33. Relationship between the responses to color and achromatic bars versus the Color Rotation Index (CRI). Units that typically respond more robustly to color bars than to achromatic bars often exhibit a higher CRI.

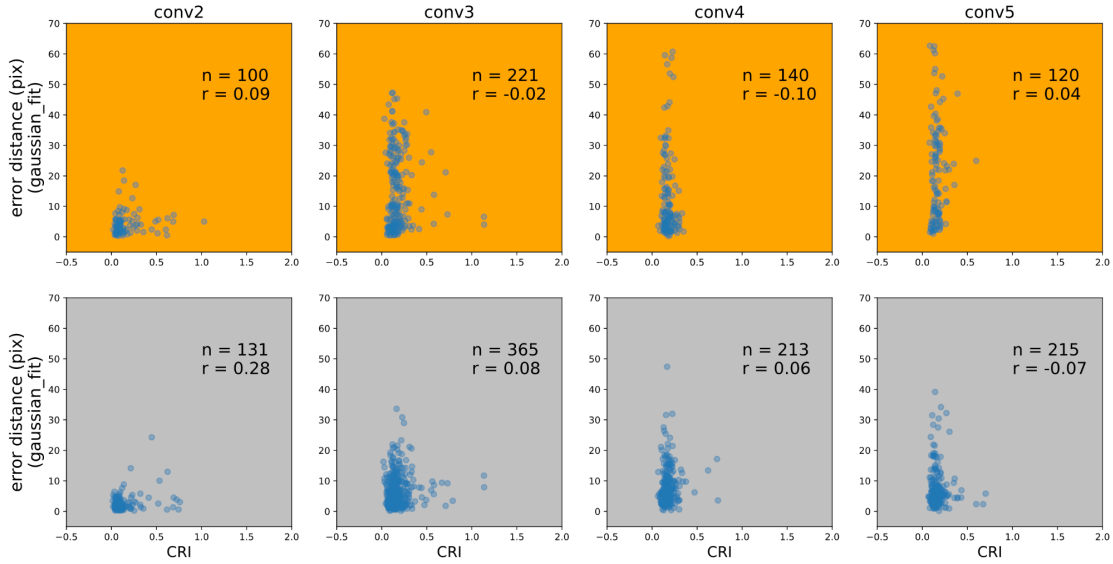

Figure 34. Relationship between the Color Rotation Index (CRI) and achromatic bar mappability. Evidently, there is no correlation between the two parameters. Thus, it can be concluded that CRI does not serve as a reliable predictor for bar mappability.

Discussion

In our study, we compared RF maps constructed with ensembles of simple bar stimuli to the ground truth maps for units in artificial networks. In all three networks we examined, we observed a trend of diminishing bar mappability, particularly in the deeper layers. In many cases, this was not due to a lack of color, leading us to suggest that many units are likely driven optimally by a complex combination of texture, color, and shape, rather than any single one of these features.

There are, however, challenges associated with using complex textures or shapes to map the ERF. In our attempts to use sinusoidal gratings and the stimulus set from Pasupathy and Connor (2001), we found difficulties in isolating which aspects of the stimuli were driving responses. Since we summed the whole shape when creating the ERF maps, we ended up with maps that were filled with artifacts from these shapes, rendering the resulting maps unusable.

An additional complexity arises when studying neural networks due to the ability of network responses to be either positive or negative. While negative responses can provide important information about a unit's receptive field, as they indicate which subregions in the receptive field can modulate the response, there isn't a direct analogue in neurophysiology given that neurons cannot have negative firing rates.

Another point of consideration is the relatively small RF in the early layers of these networks. For example, AlexNet's Conv1 has a MRF of 11 x 11. This small size makes it challenging to adequately center the bar stimulus or fit a Gaussian, prompting us to exclude these layers from our analysis. Conversely, in ResNet18, the MRF becomes larger than the input image itself beyond Conv16 (**Supplementary Figure 7**), calling for careful interpretation of results past this point. Lastly, in our examination of ResNet18, we noticed zig-zag patterns in mappability trends (**Figure 20** and **Figure 23**). The cause of these fluctuations remains unclear, but they seem to suggest that ResNet18's identity mapping may result in unusual behaviors in the ERF (**Supplementary Figure 8**).

Ultimately, our study highlights the complex nature of feature representation in CNNs and the challenges of devising accurate and comprehensive RF mapping techniques. The insights generated here should underscore the need of more accurate RF mapping in higher visual areas in both biological and artificial neural networks.

References

Amari, S. (1967). A theory of adaptive pattern classifiers. IEEE Transactions on Electronic Computers, EC-16(3), 299-307.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. PLOS ONE, 10(7), e0130140. https://doi.org/10.1371/journal.pone.0130140

Barlow, H. B. (1953). Summation and inhibition in the frog's retina. The Journal of Physiology, 119(1), 69-88.

Barlow, H. B. (1961). Possible principles underlying the transformations of sensory messages. In Sensory Communication (pp. 217-234). MIT Press.

Barlow, H. B. (1964). The coding of sensory messages. Current Problems in Animal Behaviour, 331-360.

Cavanaugh, J. R., Bair, W., & Movshon, J. A. (2002). Selectivity and spatial distribution of signals from the receptive field surround in macaque V1 neurons. Journal of Neurophysiology, 88(5), 2547-2556. doi: 10.1152/jn.00693.2001

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the Devil in the Details: Delving Deep into Convolutional Nets. In British Machine Vision Conference. Retrieved from http://www.robots.ox.ac.uk/~vgg/publications/2014/Chatfield14/chatfield14.pdf

Crick, F. (1989). The recent excitement about neural networks. Nature, 337(6203), 129-132.

Demb, J. B. (2007). Cellular mechanisms for direction selectivity in the retina. Neuron, 55(2), 179-186.

De Boer, R., & Kuyper, P. (1968). Triggered Correlation. IEEE Transactions on Biomedical Engineering, 15(3), 169-179.

De Valois, R. L., Albrecht, D. G., & Thorell, L. G. (1982). Spatial frequency selectivity of cells in macaque visual cortex. Vision Research, 22(5), 545-559.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. University of Montreal, 1341(3), 1.

Flachot, A., & Gegenfurtner, K. R. (2018). Processing of chromatic information in a deep convolutional neural network. Journal of Vision, 18(10):11, 1-15.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4), 193-202.

Güçlü, U., & van Gerven, M. A. (2015). Deep Neural Networks Reveal a Gradient in the Complexity of Neural Representations across the Ventral Stream. Journal of Neuroscience, 35(27), 10005-10014. doi: 10.1523/JNEUROSCI.5023-14.2015

Hartline, H. K. (1938). The response of single optic nerve fibers of the vertebrate eye to illumination of the retina. American Journal of Physiology. Legacy Content, 121(2), 400–415.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. The Journal of Physiology, 148(3), 574-591.
Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of Physiology, 160(1), 106-154.
Hubel, D. H., & Wiesel, T. N. (1963). Shape and arrangement of columns in cat's striate cortex. The Journal of Physiology, 165(3), 559-568.
Hubel, D. H., & Wiesel, T. N. (1965). Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. Journal of Neurophysiology, 28(2), 229-289.
Ito, M., & Komatsu, H. (2004). Representation of angles embedded within contour stimuli in area V2 of macaque monkeys. Journal of Neuroscience, 24(13), 3313-3324. doi: 10.1523/JNEUROSCI.4364-03.2004
Ivakhnenko, A. G., & Lapa, V. G. (1965). Cybernetic predicting devices. Kiev: Naukova Dumka. (in Russian)
Jones, J. P., & Palmer, L. A. (1987). An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. Journal of Neurophysiology, 58(6), 1233-1258.
Khaligh-Razavi, S. M., & Kriegeskorte, N. (2014). Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation. PLoS Computational Biology, 10(11), e1003915.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.
Kuffler, S. W. (1953). Discharge patterns and functional organization of mammalian retina. Journal of Neurophysiology, 16(1), 37-68.
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4), 541-551.
Lee, H., Ekanadham, C., & Ng, A. Y. (2008). Sparse deep belief net model for visual area V2. Advances in neural information processing systems, 20, 873-880.
Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009), 609-616.
Luo, W., Li, Y., Urtasun, R., & Zemel, R. (2016). Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. arXiv preprint arXiv:1605.06736.
Linnainmaa, S. (1970). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's Thesis (in Finnish), University of Helsinki, 6-7.
Mordvintsev, A., Olah, C., & Tyka, M. (2015). Inceptionism: Going deeper into neural networks. Google Research Blog. Retrieved from https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
Movshon, J. A., Thompson, I. D., & Tolhurst, D. J. (1978). The analysis of moving visual patterns. Experimental Brain Research, 31(1), 117-152.

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? Vision Research, 37(23), 3311-3325.
Pasupathy, A., & Connor, C. E. (2001). Shape representation in area V4: Position-specific tuning for boundary conformation. Journal of Neurophysiology, 86(5), 2505-2519.
Pospisil, D., Pasupathy, A., & Bair, W. (2018). 'Artiphysiology' reveals V4-like shape tuning in a deep network trained for image classification. eLife, 7, e38242. doi: 10.7559/eLife.38242
Rosenblatt, F. (1962). Principles of Neurodynamics. Washington, D.C.: Spartan Books.
Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. 2017 IEEE International Conference on Computer Vision (ICCV), 618-626. https://doi.org/10.1109/ICCV.2017.74
Sherrington, C. S. (1906). The integrative action of the nervous system. New Haven: Yale University Press.
Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. Workshop at International Conference on Learning Representations (ICLR).
Simonyan, K. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034 [cs.CV].
Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for Simplicity: The All Convolutional Net. arXiv:1412.6806 [cs.LG].
Walker, G. A., Ohzawa, I., & Freeman, R. D. (2000). Suppression outside the classical cortical receptive field. Visual Neuroscience, 17(3), 369-379.
Zeiler, M. D., & Fergus, R. (2013). Visualizing and understanding convolutional networks. arXiv preprint arXiv:1311.2901.
Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning Deep Features for Discriminative Localization. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2921-2929. https://doi.org/10.1109/CVPR.2016.319
Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Object Detectors Emerge in Deep Scene CNNs. arXiv preprint arXiv:1412.6856.
Zhou, H., Friedman, H. S., & von der Heydt, R. (2000). Coding of border ownership in monkey visual cortex. Journal of Neuroscience, 20(17), 6594-6611. doi: 10.1523/JNEUROSCI.20-17-06594.2000

Appendices

Supplementary Figures

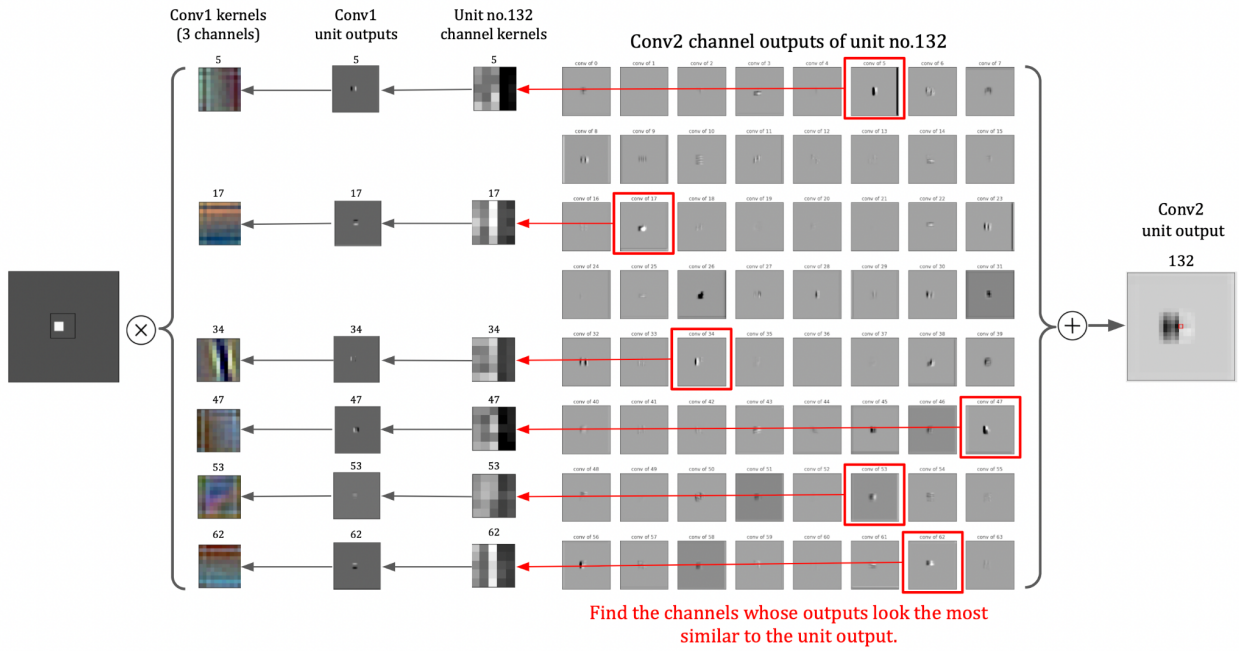

Figure 1. Investigating the Mechanism of Conv2 Unit no. 132 in AlexNet. We began by examining the output of Unit no. 132 of Conv2, focusing on the channel outputs comprising it. The red box within the unit output feature map emphasizes the unit situated at the spatial center. We identified the channel outputs with the highest similarity to the unit output, allowing us to trace back to the channel kernels of this unit. In this case, the right two columns of each channel all have negative weights. As each channel originated from a Conv1 unit, we analyzed the activations and kernels of those Conv1 units. These units were found to be filters detecting edges of varying orientations and contrast polarities.

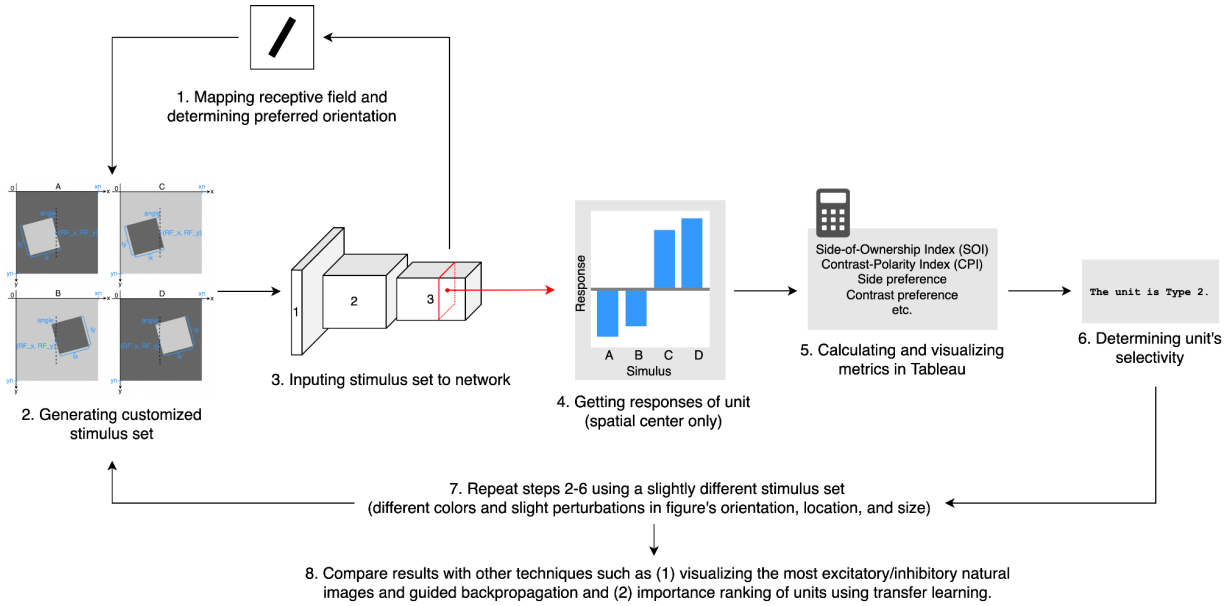

Figure 2. The 'Artiphysiology' Method Applied to Border Ownership Research. Bair Lab developed a methodology called 'Artiphysiology' to study artificial neural networks in a manner similar to neurophysiological approaches (Pospisil et al., 2018). The process began by mapping the receptive fields of AlexNet units, which allowed for the characterization of each unit based on its preferences for stimulus orientation, size, and color. Customized border-ownership stimuli were then generated according to these preferences. The stimuli were presented to the neural network, and the responses were used to calculate a set of metrics similar to those used by Zhou et al. (2000). This step was repeated with slight variations in stimulus location, orientation, size, and color to ensure that the observed visual selectivity remained invariant to these transformations.

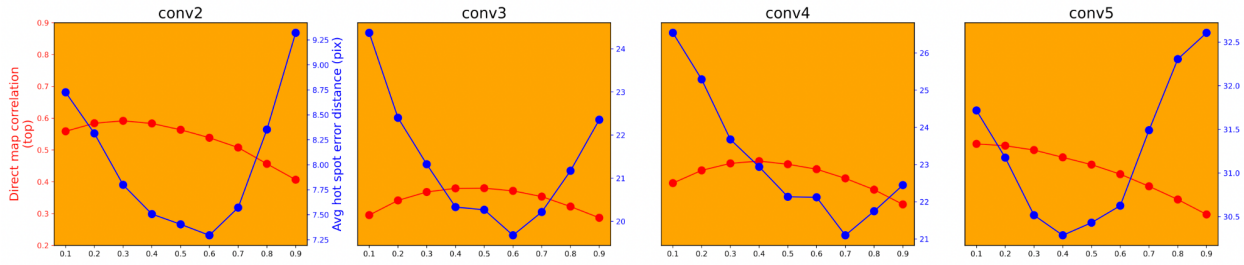

Figure 3. Determining response threshold for bar maps. We tested response thresholds from 0.1 to 0.9 in increments of 0.1 when generating the bar maps. The resulting bar maps were then compared to the ground-truth guided backpropagation maps using two mappability metrics: Pearson correlation coefficient (red curve) and hotspot peak (blue curve). The curves represent the average metric values for Conv2 through Conv5 of AlexNet. The data suggests an optimal response threshold within the range of 0.4 to 0.7.

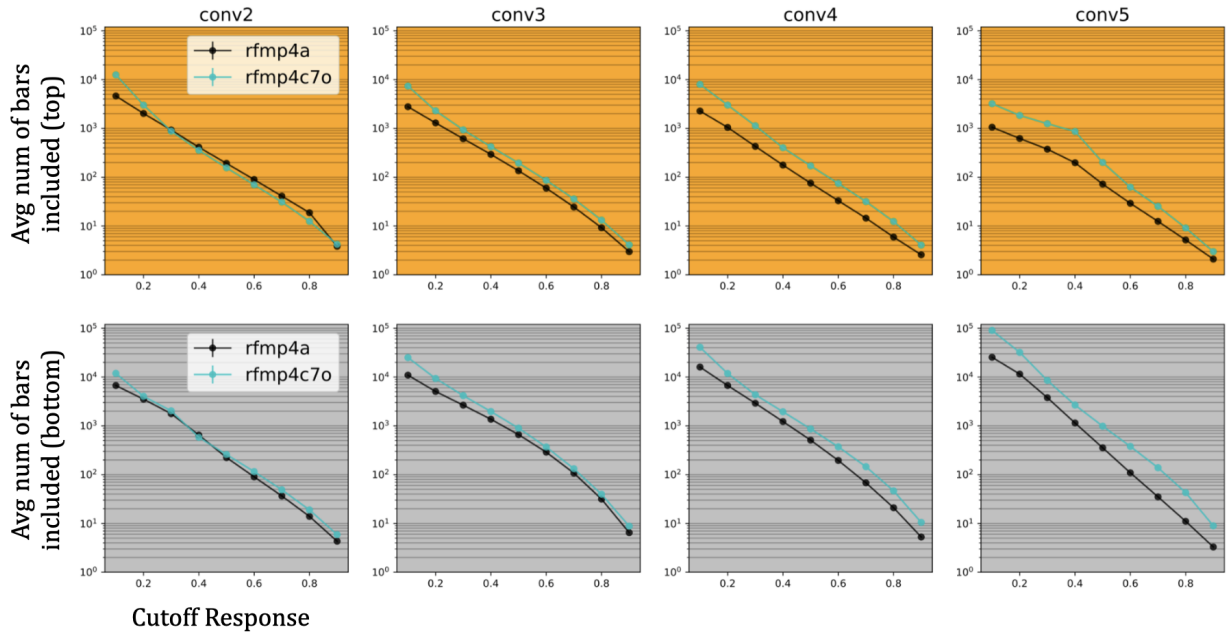

Figure 4. Relationship between the average number of bars included in the bar maps and the cutoff response when the y-axis is transformed on a logarithmic scale. The curve is approximately linear and slightly concave, indicating that the distribution of responses is right-skewed. A small number of bars (in single digits) have responses above 90% of the maximum responses. Interestingly, in deeper layers, an offset is observed between the color bar curve (cyan) and the achromatic bar curve (black). This suggests that the color bars follow a similarly right-skewed distribution as the achromatic bars, with the number of color bars included being proportional to the number of achromatic bars. We use abbreviations to denote the type of bar map: rfmp4a for achromatic bars and rfmp4c7o for color bars.

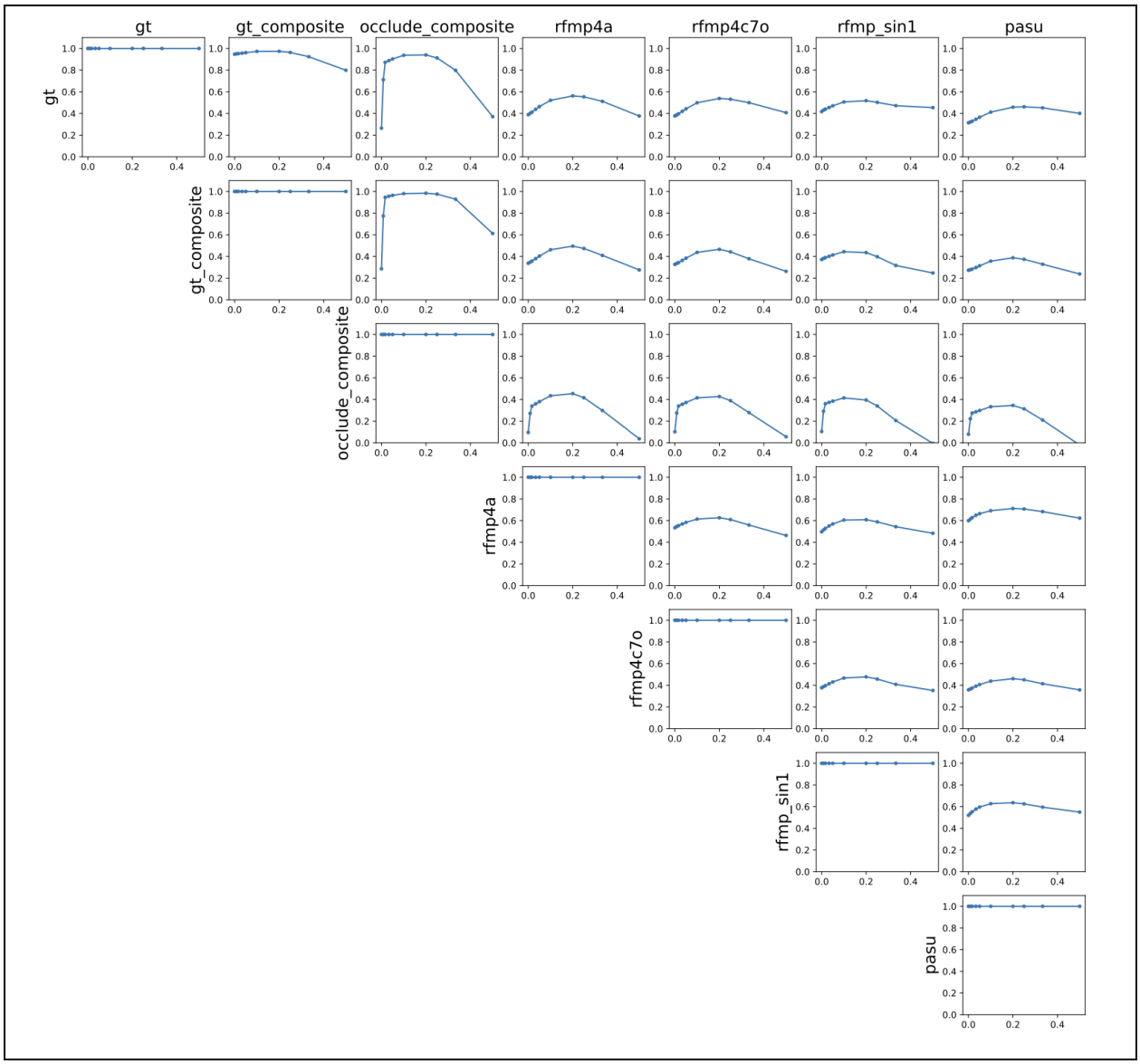

Figure 5. Evaluation of various Gaussian smoothing standard deviations (SD) for map comparisons. We tested SD values of [0, 1/120, 1/60, 1/30, 1/20, 1/10, 1/5, 1/4, 1/3, 1/2] M, where M represents the MRF size. The vertical axis displays the average Pearson correlation coefficient between the two maps for a single layer (Conv3 of AlexNet in this case). Map abbreviations are as follows: gt (ground truth guided backpropagation - top images), gt_composite (ground truth guided backpropagation - top + bottom images), occlude_composite (occlusion map - top + bottom images), rfmp4a (achromatic bar map - top bars), rfmp4c7o (color bar map - top bars), rfmp_sin1 (sinusoidal gratings - top gratings), and pasu (Pasupathy shapes - top shapes).

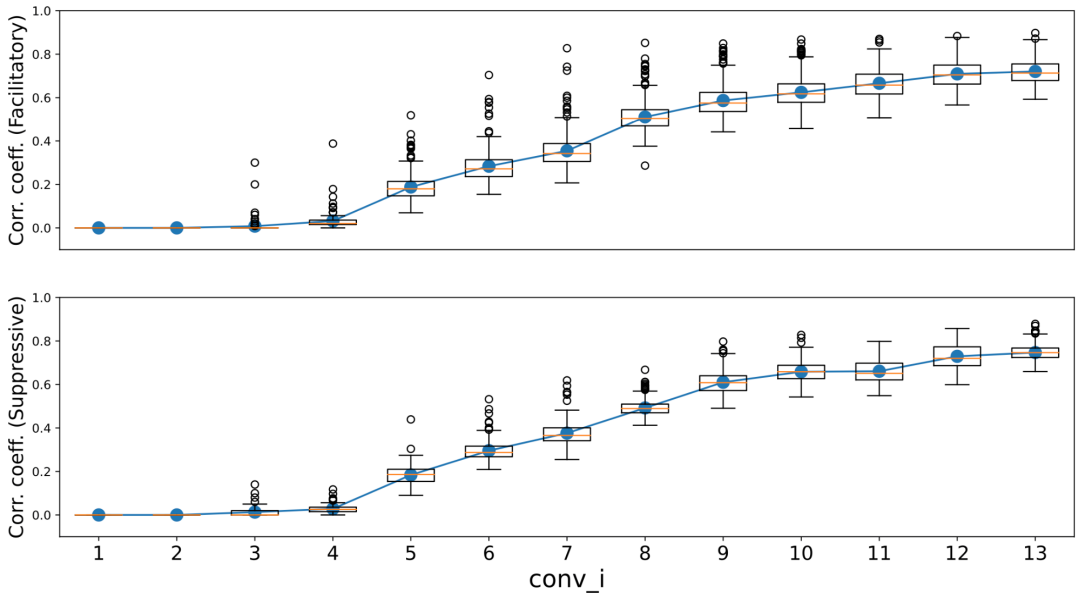

Figure 6. Sparsity of ground-truth (guided backpropagation) maps of AlexNet, where sparsity is defined as the proportion of pixels with values below an absolute tolerance of 0.01. The sparsity gradually increases as we move into deeper layers.

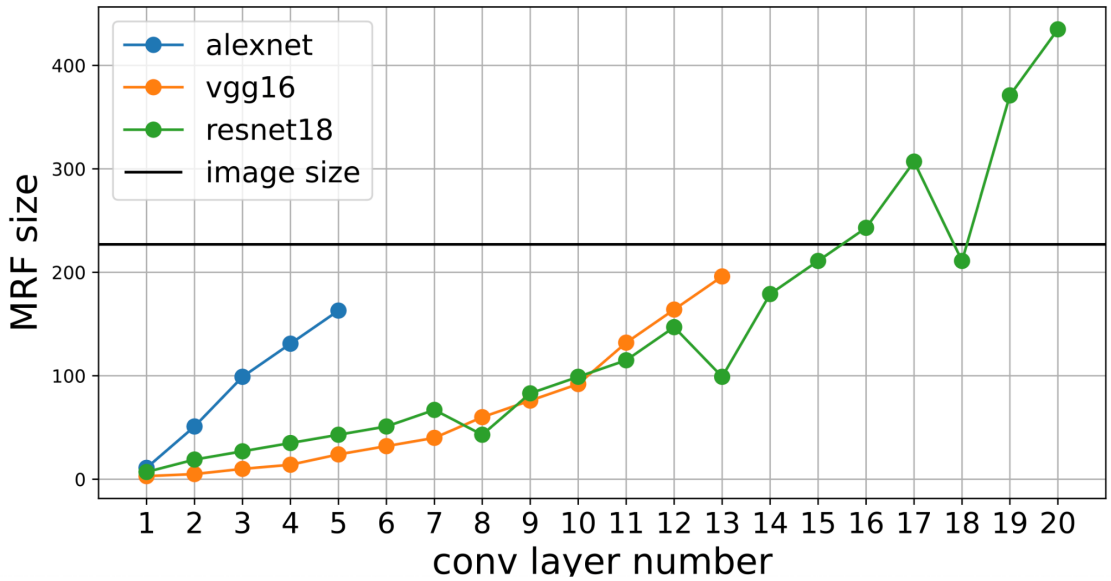

Figure 7. Maximum receptive field (MRF) sizes for the three models. In ResNet18, dips in MRF sizes at Conv8, Conv13, and Conv18 can be attributed to these layers being convolutional layers on the skip connections, which are shown in **Supplementary Figure 8**. Note that in ResNet18, Conv16, Conv17, Conv19, and Conv20 have MRF size greater than 227 (black horizontal line), which is the size of the input image.

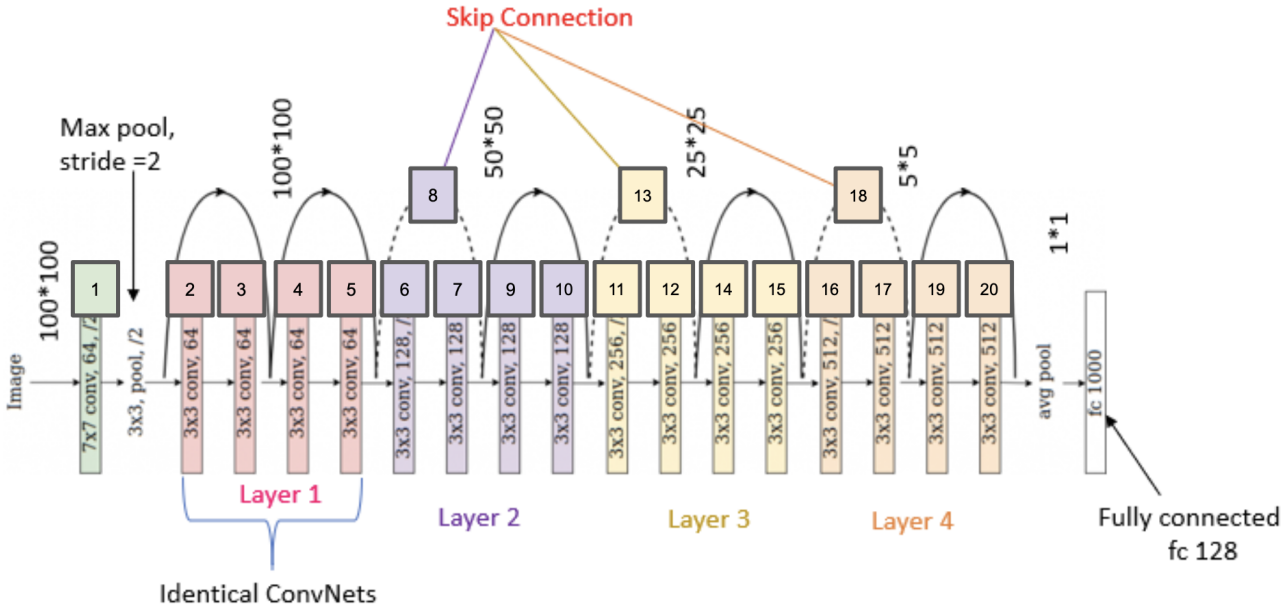

Figure 8. Layer indexing in ResNet18. Due to the presence of skip connections, the indexing of convolutional layers becomes slightly more arbitrary. We followed the flow of image data through the layers (using PyTorch's hooks) to index the convolutional layers such that the layers on the skip connections are indexed last in each block (image modified from He, et al., 2015).

Algorithms expressed using Python-like pseudocodes

Neurophysiologically inspired RF mapping methods

Achromatic bars:

```
# Present bars to truncated model.
responses = np.zeros((num_units, num_bars))
i = 0
for batch in bars:
    responses[:, i:i + batch_size] = truncated_model(batch)
    i += batch_size

# Make bar maps.
RESPONSE_THRESHOLD = 0.5
top_bar_maps = np.zeros((num_units, map_height, map_width))
bottom_bar_maps = np.zeros((num_units, map_height, map_width)
for unit in range(num_units):
    bar_i_sorted = argsort(responses[unit], descending=True)

    # Make top bar maps.
    max_response = responses[unit, bar_i_sorted[0]]
    response = max_response
    num_top_bars = 0
    while response > max(max_response * RESPONSE_THRESHOLD, 0):
        bar_i = bar_i_sorted[num_top_bars]
        response = responses[unit, bar_i]
        bar_location_map = make_bar(bar_parameters[bar_i],
                                    bar_color=1.0, background_color=0.0)
        top_bar_maps[unit] += bar_location_map * response
        num_top_bars += 1

    # Make bottom bar maps.
    bar_i_sorted = bar_i_sorted[::-1]  # make descending
    min_response = responses[unit, bar_i_sorted[0]]
    response = 0
    num_bottom_bars = 0
    while response < min(min_response * RESPONSE_THRESHOLD, 0):
        bar_i = bar_i_sorted[num_buttom_bars]
        response = responses[unit, bar_i]
        bar_location_map = make_bar(bar_parameters[bar_i],
                                    bar_color=1.0, background_color=0.0)
        bottom_bar_maps[unit] += bar_location_map * response
        num_bottom_bars += 1
```

Color bars (The differences with achromatic bar algorithm are highlighted in red)

```
# Present bars to truncated model.
responses = np.zeros((num_units, num_bars))
i = 0
for batch in bars:
    responses[:, i:i + batch_size] = truncated_model(batch)
    i += batch_size

# Make bar maps.
RESPONSE_THRESHOLD = 0.5
top_bar_maps = np.zeros((num_units, map_height, map_width))
bottom_bar_maps = np.zeros((num_units, map_height, map_width)
for unit in range(num_units):
    bar_i_sorted = argsort(responses[unit], descending=True)

    # Make top bar maps.
```

```
    max_response = responses[unit, bar_i_sorted[0]]
    response = max_response
    num_top_bars = 0
    while response > max(max_response * RESPONSE_THRESHOLD, 0):
        bar_i = bar_i_sorted[num_top_bars]
        response = responses[unit, bar_i]
        bar_color = bar_colors[bar_i]
        if bar_color == (-1,-1,-1):  # if bar is black
            bar_location_map = make_bar(bar_parameters[bar_i],
                                        bar_color=(1,1,1), background_color=0.0)
        else:
            bar_color[bar_color < 0] = 0  # ReLU
            bar_location_map = make_bar(bar_parameters[bar_i],
                                        bar_color=bar_color, background_color=0.0)
        top_bar_maps[unit] += bar_location_map * response
        num_top_bars += 1

    # Make bottom bar maps.
    bar_i_sorted = bar_i_sorted[::-1]  # make descending
    min_response = responses[unit, bar_i_sorted[0]]
    response = 0
    num_bottom_bars = 0
    while response < min(min_response * RESPONSE_THRESHOLD, 0):
        bar_i = bar_i_sorted[num_buttom_bars]
        response = responses[unit, bar_i]
        bar_color = bar_colors[bar_i]
        if bar_color == (-1,-1,-1):  # if bar is black
            bar_location_map = make_bar(bar_parameters[bar_i],
                                        bar_color=(1,1,1), background_color=0.0)
        else:
            bar_color[bar_color < 0] = 0  # ReLU
            bar_location_map = make_bar(bar_parameters[bar_i],
                                        bar_color=bar_color, background_color=0.0)
        bottom_bar_maps[unit] += bar_location_map * response
        num_bottom_bars += 1
```

“Ground truth” methods

Natural Images

```
# Get the maximum response and the corresponding spatial location.
max_responses = np.zeros((num_images, num_units))
max_locations = np.zeros((num_images, num_units, 2))
for image_index in range(num_images):
    for unit in range(num_units):
        truncated_model = model[:conv_layer_index+1]
        output = truncated_model(images[image_index])
        imagewise_max_response = np.max(output)
        y, x = np.unravel_index(np.argmax(output))
        max_responses[image_index, unit] = imagewise_max_response
        max_locations[image_index, unit, 0] = y
        max_locations[image_index, unit, 1] = x

# Sort according to responses
sorted_image_indices = np.zeros((num_images, num_units))
sorted_max_locations = np.zeros((num_images, num_units, 2))
for unit in range(num_units):
    sorted_image_indices[:, unit] = np.argsort(max_responses[:, unit])
    sorted_max_locations[:, unit] = max_locations[sorted_image_indices[:, unit], unit]

# Example: show the top 1 patch of unit no.23.
image_index = sorted_image_indices[-1, 23]
y, x = sorted_max_locations[-1, 23]
mask = back_project_location(y, x)
show(images[image_index][mask])
```

Guided backpropagation

```
# Define guided backpropagation
def guided_backprop(image_patch, truncated_model, unit, location):
    # Forward pass
    forward_masks = []
    x = image_patch.copy()
    for layer in truncated_model.layers:
        x = layer(x)
        if isinstance(layer, ReLU):
            new_forward_mask = x > 0
            forward_masks.append(new_forward_mask)

    # Target only the location we want
    x = x[unit, location]

    # Backward pass
    gradient_image = torch.zeros_like(image_patch)
    grad = torch.ones_like(x)
    for layer in reversed(truncated_model.layers):
        if isinstance(layer, ReLU):
            grad = grad * (x > 0)
            grad = grad * forward_masks.pop()
        grad = layer.backward(grad)
    gradient_image = grad
    gradient_image = np.mean(gradient_image, color_channel)
    return np.absolute(gradient_image)

# Compute guided backprop of top 100 patches
guided_backprop_results = np.zeros((num_units, rf_height, rf_width))
for unit in range(num_units):
    for rank in range(100):
        image_patch = image_patches_sorted_descending[rank, unit]
        location = max_locations_sorted_descending[rank, unit]
        gradient_image = guided_backprop(image_patch, model[:layer+1], unit, location)
        gradient_image = np.mean(gradient_image, color_channel)
        gradient_image = np.absolute(gradient_image)
        guided_backprop_results[unit] += gradient_image
```

Gradient ascent

```
LEARNING_RATE = 0.1  # named for historical reason, not actually “learning”
gradient_ascent_results = np.zeros((num_units, rf_height, rf_width))

# Compute gradient ascent images for every unit.
for unit in range(num_units):
    # Using the top-10 patches as initial images
    for rank in range(10):
        image_patch = image_patches_sorted_descending[rank, unit]
        x = image_patch.copy()
        i = 0
        converge = False
        while i < max_iter and not converge:
            responses = truncated_model(x)
            center_response = responses[unit, center_y, center_x]
            grad = torch.autograd.grad(center_response, x)[0]
            x += LEARNING_RATE * grad
            i += 1

        x -= image_patch
        gradient_image = np.mean(x, color_channel)
        gradient_image = np.absolute(x)
        gradient_ascent_results[unit] += gradient_image
```

Occlusion

```
# Occlude top-100 patches and get the absolute difference in response
```

```
discrepancy_map = np.zeros((units, rf_size, rf_size))
for unit in range(num_units):
    # Using the top-100 patches as initial images
    for rank in range(100):
        image_patch = image_patches_sorted_descending[rank, unit]
        original_response = truncated_model(image_patch)[unit, center_y, center_x]
        for j in range(0, rf_size, occluder_stride):
            for i in range(0, rf_size, occluder_stride):
                if i or j out of range:
                    continue

                x = image_patch.copy()
                x[j:j+occluder_size,i:i+occluder_size] = torch.rand((3,rf_size,rf_size))*2 - 1
                response = truncated_model(x)[unit, center_y, center_x]
                response_abs_diff = abs(original_response - response)
                discrepancy_map[unit, j+rf_size//2, i+rf_size//2] += response_abs_diff
```

Comparing two maps: Mappability metrics

Pearson correlation

```
# Smooth map
SIGMA = RF_SIZE / 30
for color in range(3):
    ground_truth_map[:,:,color] = gaussian_filter(ground_truth_map[:,:,color], sigma=SIGMA)
    bar_map[:,:,color] = gaussian_filter(bar_map[:,:,color], sigma=SIGMA)

# Normalize map
ground_truth_map /= ground_truth_map.max()
bar_map /= bar_map.max()

# Remove color channel
def remove_color(rf_map):
    if len(rf_map.shape) == 3:
        rf_map = np.mean(rf_map, axis=2)
    return rf_map

ground_truth_map = remove_color(ground_truth_map)
bar_map = remove_color(bar_map)
```

Intersection Over Union (IoU)

```
# Smooth map
SIGMA = RF_SIZE / 30
for color in range(3):
    ground_truth_map[:,:,color] = gaussian_filter(ground_truth_map[:,:,color], sigma=SIGMA)
    bar_map[:,:,color] = gaussian_filter(bar_map[:,:,color], sigma=SIGMA)

# Normalize map
ground_truth_map /= ground_truth_map.max()
bar_map /= bar_map.max()

# Remove color channel
def remove_color(rf_map):
    if len(rf_map.shape) == 3:
        rf_map = rf_map.max(axis=2)
    return rf_map

ground_truth_map = remove_color(ground_truth_map)
bar_map = remove_color(bar_map)

# Compute IoU
IOU_THRESHOLD = 0.5
def compute_iou(map1, map2):
```

```
    binary_map1 = map1 > IOU_THRESHOLD
    binary_map2 = map2 > IOU_THRESHOLD
    intersection = np.sum(np.logical_and(binary_map1, binary_map2))
    union = np.sum(np.logical_or(binary_map1, binary_map2))
    if math.isclose(union, 0.0):
        return 0
    return intersection/union

iou = compute_iou(ground_truth_map, bar_map)
```

Distance between centers

```
# Remove color channel
if len(rf_map.shape) == 3:
    rf_map = rf_map.max(axis=2)

# Smooth map
SIGMA = RF_SIZE / 30
if center_finding_method == “com”:
    if map_name == “occlude”:  # For CoM, only smooth the occlude map.
        rf_map = gaussian_filter(rf_map, sigma=SIGMA)
else:
   rf_map = gaussian_filter(rf_map, sigma=SIGMA)

# Normalize map
ground_truth_map /= ground_truth_map.max()
bar_map /= bar_map.max()

# Locating RF center:
center_coordinates = None

# Center of Mass (CoM)
if center_finding_method = “com”:
    cy, cx, radius10 = compute_com(rf_map, 0.1)
    _ , _ , radius50 = compute_com(rf_map, 0.5)
    _ , _ , radius90 = compute_com(rf_map, 0.9)
    center_coordinates = [cy, cx]

elif center_finding_method = “hotspot”:
    center_coordinates = np.unravel_index(np.argmax(rf_map), rf_map.shape)

elif center_finding_method = “gaussian”:
    params = gaussian_fit(rf_map)
    center_coordinates[0] = params[‘cy’]
    center_coordinates[1] = params[‘cx’]

    # compute fraction of explained variance:
    fit_map = make_2d_gaussian(params)
    residual_var = variance(fit_map.flatten() - rf_map.flatten())
    rf_map_var = variance(rf_map.flatten())
    fxvar = 1 - (residual_var / rf_map_var)
```

Discrepancy analysis: Diagnostic metrics

Color rotation index (CRI)

```
# Get a random set of 1000 images
img_list = random.choices(range(50000), k=1000)

# Compute CRI of all units
responses_to_original_image = np.zeros((1000, num_units))
SD_cr_list = np.zeros((1000, num_units))
img_count = 0
for img_idx in img_list:
    img = images[img_idx]
    temp_responses = np.zeros((6, num_units))
```

```
    # Get all permutations
    for i, (ri, gi, bi) in enumerate(permutations([0,1,2])):
        permutated_image = torch.zeros_like(img)
        permutated_image[ri] = img[0]
        permutated_image[gi] = img[1]
        permutated_image[bi] = img[2]
        res = truncated_model(permutated_image)[:, center_y, center_x]
        temp_responses[i] = res
        if ri == 0 and gi == 1 and bi == 2:
            responses_to_original_image[img_count] = res
    SD_cr_list[img_count] = np.std(temp_responses, axis=0)
    img_count += 1
cri_list = np.mean(SD_cr_list, axis=0) / np.std(responses_to_original_image, axis=0)
# cri_list has shape (num_units,)
```