# Interpretable Machine Learning with Applications in Neuroscience

*Reza Abbasi Asl*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2018-50
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-50.html

May 11, 2018

**Interpretable Machine Learning with Applications in Neuroscience**

by

Reza Abbasi Asl

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

and the Designated Emphasis

in

Communication, Computation and Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Bin Yu, Chair
Professor Kannan Ramchandran
Professor Jack L. Gallant

Spring 2018

**Interpretable Machine Learning with Applications in Neuroscience**

## Abstract

Interpretable Machine Learning with Applications in Neuroscience

by

Reza Abbasi Asl

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Bin Yu, Chair

In the past decade, research in machine learning has been principally focused on the development of algorithms and models with high predictive capabilities. Models such as convolutional neural networks (CNNs) achieve state-of-the-art predictive performance for many tasks in computer vision, autonomous driving, and transfer learning. However, interpreting these models remains a challenge, primarily because of the large number of parameters involved.

In this thesis, we investigate two regimes based on (1) compression and (2) stability to build more interpretable machine learning models. These regimes will be demonstrated in a computational neuroscience study. In the first part of the thesis, we introduce a greedy structural compression scheme that prunes filters in a trained CNN. To do this, we define a filter importance index equal to the classification accuracy reduction (CAR) of the network after pruning that filter (similarly defined as RAR for regression). CAR achieves state-of-the-art classification accuracy compared to other filter pruning schemes. Furthermore, we demonstrate the interpretability of CAR-compressed CNNs by showing that our algorithm prunes filters with visually redundant functionalities such as color filters.

In the second part of this thesis, we introduce DeepTune, a stability-driven visualization and interpretation framework for CNN-based models. DeepTune is used to characterize biological neurons in the V4 area of the primate visual cortex. V4 is a large retinotopically-organized area of the visual cortex located between the primary visual cortex and high-level areas in the inferior temporal lobe. V4 neurons have highly nonlinear response properties and it is notoriously difficult to construct quantitative models that accurately describe how visual information is represented in V4. To better understand the filtering properties of these neurons, we study recordings from 71 well isolated cells stimulated with 4000-12000 static grayscale natural images collected by the Gallant Lab

at UC Berkeley. Our CNN-based models of V4 neurons achieve state-of-the-art accuracy in predicting neural spike rates in a hold-out validation set (average predictive correlation of 0.53 for 71 neurons). Then, we employ our DeepTune stability-driven interpretation framework and discover that the V4 neurons are tuned to a remarkable diversity of textures (40% of the neurons), contour shapes (30% of the neurons), and complex patterns (30% of the neurons). Most importantly, these smooth DeepTune images provide testable naturalistic stimuli for future experiments on V4 neurons.

In the final part of this thesis, we study the application of CAR and RAR compressed CNNs in modeling V4 neurons. Both CAR and RAR compression give rise to a new set of simpler models for V4 neurons with similar accuracy to existing state-of-the-art models. For each of the accurate CAR and RAR compressed models of V4 neurons (up to 90% compression rate), the smooth DeepTune images are stable and exhibit similar patterns to the uncompressed model's consensus DeepTune image. Our results suggest, to some extent, that these CNNs resemble the structure of the primate brain.

To my family

# Contents

# List of Figures

# List of Tables

# Acknowledgments

During my Ph.D. studies at Berkeley, I have been blessed to learn and grow with the help of numerous talented researchers and professionals. I am greatly thankful to these mentors, colleagues and friends who helped me to become who I am today.

First and foremost, I would like to thank my principal advisor, Bin Yu. She has been an amazing mentor, teacher, and most importantly a role model for me during the last five years. She has kindly helped me through the ups and downs of my PhD studies, guiding me to improve and become ready for my future career. I owe her a great debt of gratitude for all of her support both as an advisor and as a friend.

I am also thankful for the mentoring of Jack Gallant, who has been a great person to learn from and work with. He has always been a true inspiration for me to learn more about neuroscience and collaborating with him and his group has been a unique opportunity for me to improve my understanding of vision and the brain. I am also grateful to Kannan Ramchandran and Bruno Olshausen, other members of my thesis and qualifying examination committee. Their helpful feedback and comments during the last few years have been a valuable source of improvement to this thesis.

Chapter 3 of this thesis is the outcome of a collaboration with Yuansi Chen, Adam Bloniarz, Michael Oliver, Ben Willmore, Jack Gallant, and Bin Yu. Yuansi has been an awesome person to work with. I really enjoyed working and collaborating with him during these years, especially discussing research problems through our day-long meetings. I am also grateful to Adam who was a great mentor, coaching me when I first arrived in Berkeley. I appreciate all the efforts of Ben and Michael in collecting the data discussed in this thesis and their helpful suggestions during the analysis and the writing of these papers.

I am also thankful to former and current members of Yu Group who have been a great source of support during my last 5 years at Berkeley. I am grateful to all of them, particularly Karl Kumbier, Sujayam Saha, Rebecca Barter, Simon Walter, Jaime Murdoch, Chandan Singh, Sören Künzel, Raaz Dwivedi, Yu Wang, Nicholas Altieri, Christine Kuang, Hanzhong Liu, Sumanta Basu, Sivaraman Balakrishnan, Siqi Wu, and Hongwei Li.

Aside from my research at UC Berkeley, during my Ph.D., I also had two valuable internship experiences. I spent one summer under the supervision of Mitya Chklovskii at the Simons Foundation's Flatiron Institue in New York. I learned a lot from him and Cengiz Pehlevan and I am thankful to them for being such great mentors. I also spent summer of 2017 at Neuralink Corp. and was fortunate to work with Philip Sabes and Paul Merolla on several problems in brain-computer interfaces.

I had the opportunity to serve as Graduate Student Instructor (GSI) for three semesters at Berkeley: two semester for the course EE120: Signals and Systems with Thomas Courtade and one semester for the course EE20: Interpretation of Signals and Systems with Babak Ayazifar and Avideh Zakhor. I am extremely thankful to the these awesome instructors and mentors for teaching how to teach.

I have been very fortunate to be supported financially for the last five years by The Center for Science of Information (CSoI), a US NSF Science and Technology Center under grant agreement CCF-0939370. I really enjoyed being a part of this center and attending many fruitful site visits and workshops at Purdue.

I am also thankful to the staff members in EECS and Statistics Departments who have been always extremely helpful during the last 5 years. Special thanks goes to La Shana Porlaris and Shirley Salanio for all their help.

During the last 5 years, I have been fortunate to know, interact and learn from members of WiFo and BLISS: Payam Delgosha, Ramtin Pedarsani, Fanny Yang, Nihar Shah, Rashmi K.V, Kangwook Lee, Sameer Pawar, Venkatesan Ekambaram, Sreeram Kannan, Po-Ling Loh, Varun Jog, Gireeja Ranade, Kate Harrison, Giulia Fanti, Ilan Shomorony, Vasuki Narasimha Swamy, Yuting Wei, Vidya Muthukumar, Ashwin Pananjady, Orhan Ocal, Dong Yin, Sang Min Han, Govinda Kamath, and Mahdi Soltanolkotabi.

Additionally, I am also grateful to other people from the Gallant Lab: James Gao, Alex Huth, Fatma Imamoglu, Mark Lescroart, Lydia Majure, Leila Wehbe, Robert Gibboni, Natalia Bilenko, Anwar Nunez-Elizalde, Storm Slivkoff, Sara Popham, and Tianjiao Zhang, and Dustin Stansbury for all the great discussions that have inspired me to think about challenging problems in neuroscience.

Finally and most importantly, I would like to thank my family, Maryam, mom, dad, Ladan, Mohammad Behgam, Masoumeh, Fatemeh, Sara, and Hootan for their unconditional love and support. I undoubtedly could not have achieved this without them.

# Chapter 1

# Overview

Recent developments in machine learning research such as artificial deep neural networks (DNN) have achieved cutting-edge performance for many tasks in Artificial Intelligence. Areas such as computer vision, autonomous driving and natural language and speech processing have been the focus of this developments in the past decade. However, the substantial number of weights in these accurate machine learning tools has made it challenging to humanly interpret or investigate these models for possible domain knowledge gain. While there is no unified technical definition of interpretability in the literature, interpretablity in machine learning is loosely described as the ability to explain or to present the decisions made by the model in understandable terms to a human [18]. Interpretability is typically studied from one of two perspectives [39, 10, 22, 41, 17]. The first is algorithmic interpretablity and transparency of the algorithmic learning. The other is post-hoc interpretability and explanation of the learned model using tools such as visualization. The first perspective attempts to answer the question of how the model learns and works, while the second perspective describes the output (prediction, etc.) from learned parameters without explaining the algorithmic learning. Although the statistics community has a long history of studying transparent and elegant models such as linear regression, neither of the two perspectives have been sufficiently investigated for the state-of-the-art machine learning algorithms and models.

Our main goal throughout this thesis is to develop and evaluate interpretable machine learning tools using stability principle [73] and ideas from applied statistics. In particular, we focus on stability-driven post-hoc interpretation and intelligibility of machine learning models. Our main goal is to build and to interpret computational models using large volumes of data in domains such as neuroscience. In these cases, model interpretability is an essential part of ensuring that the conclusions drawn from the model will be considered trustworthy by the scientific community. In the field of computational neuroscience, we

will focus on understanding the visual pathway of primates' brains. Machine learning models such as convolutional neural networks (CNNs) inspired by the physiology of the visual cortex have been widely used to predict neural spike rates and to explain representations in primate's visual cortex [7]. Although these models achieve state-of-the-art performance in predicting neural spike rates, a reliable interpretation of these models is necessary to make any scientific conclusion regarding the functionality of neurons. An interpretable machine learning algorithm will help pave the road towards important discoveries in domains such as computational neuroscience, vision science and biology. In the following paragraphs, I will summarize the main contributions of this thesis.

**Structural compression of convolutional neural networks** Huge numbers of parameters in deep CNNs often make it difficult to interpret these network. From the perspective of post-hoc interpretability, a network with fewer parameters is easier to visualize and explain to a human interpreter. Therefore, compression has a natural role in building more interpretable models. CNNs are often visualized using graphical explanation of their filters [74]. Thus to increase the interpretability, a compression method should decrease the number of filters. This is different from classical approaches in compression of CNNs where individual weights are pruned and quantized [23]. To fill in the gap, we have introduced a greedy structural compression scheme that prunes filters in a trained CNN [2, 1]. We first quantify the importance of each filter by the classification accuracy reduction (CAR) of the network after pruning that filter. Then, the filters are iteratively pruned based on the CAR index. Our algorithm achieves remarkably higher classification accuracy (up to 25% for AlexNet [33]) compared to the best benchmark filter pruning schemes. When combined with further weight pruning, coding and quantization, the CAR algorithm reduces the size of individual convolutional layers in AlexNet by a factor of 8 to 15, while achieving close to original accuracy through retraining. More importantly, structural compression leads to a more interpretable CNN. We demonstrated the interpretability of CAR-compressed CNNs by showing that CAR algorithm prunes filters with visually redundant functionalities such as color filters. Additionally, when working on our structural compression, we discovered that using a variant of our CAR index, it is possible to present a verbal and graphical interpretation of each filter in a CNN. We were able to quantify the importance of each image category to each CNN filter. By identifying the most and the least important class labels for each filter, we presented a meaningful post-hoc interpretation of each filter.

**Interpretable computational models of neurons in the visual area V4** The Yu research group, that I am part of, has had a long-term collaboration with Jack Gallant's lab at UC Berkeley, one of the leading neuroimaging and neurophysiology labs in the world.

This collaboration has led to several pioneering works in computational neuroscience. Among them is the first visual experience decoder from brain activity [46] which was declared one of the top 50 best inventions of 2011 by TIME magazine. By virtue of this collaboration, I had the opportunity to work closely with remarkable neuroscientists on datasets varied from measurements of single neurons to fMRI recordings of the whole brain.

Together with my collaborators at Yu Group and Gallant lab, we have developed an interpretable model for single neurons in area V4 of the primate's visual cortex. Intermediate areas of visual pathway including the V4 area are known to play an important role in visual object recognition. While the spiking patterns of neurons in the early visual cortex (V1 area) can be fairly accurately modeled using a few parameters such as spatial and frequency tunings [8], these methods fail to generalize to area V4, due to its highly nonlinear response properties [56]. To uncover the functionality of these neurons, we have studied recordings from 71 neurons in the cortical area V4 of two macaque monkeys being shown thousands of static natural images. CNN-based predictive models of neuron spike rates achieve state-of-the-art performance for V4 neurons. To build these models, we train a CNN to accurately classify natural images. Then for each stimulus image, a feature-set is extracted by feeding the image forward through several layers of this network. Finally, a linear model (Ridge or Lasso) predicts the neuron spike rates from this feature-set. By varying the CNN architecture and number of layers, 18 distinct and accurate models emerge from this setting. To visualize the pattern selectivity and tuning of each neuron in V4, we introduce "DeepTune", a statistical framework for stability-driven [73] interpretation of our 18 models. DeepTune presents a visualization of neural tuning and pattern selectivity using model-based naturalistic optimized images. Each V4 neuron's spatial pattern selectivity is characterized by DeepTune. Through our stability-driven interpretation framework, we discover that the V4 neurons are tuned to a remarkable diversity of textures (40% of the neurons) and shapes such as contours (30% of the neurons). We also demonstrate that close to half of the V4 neurons exhibit strong suppressive tuning. Most importantly, these smooth DeepTune images provide testable naturalistic stimuli for future experiments on V4 neurons.

Through this visualization, we show that V4 neurons are selective to a remarkable diversity of shapes such as curvatures, textures and V1-like gratings.

**Compressed models of neurons in visual area V4** The computational models for V4 neurons based on pre-trained CNNs often have thousands of parameters. The large number of parameters in these models makes it hard to interpret them. We study the post-hoc interpretability of these models. To simplify these models, and consequently to make them more interpretable, we use CAR-compressed CNNs in chapter 4 to build

predictive models of V4 neurons. The modeling procedure is similar to chapter 3, but instead of full CNN, the CAR-compressed version is employed to predict spike rate of V4 neurons. We show that CAR-compressed networks achieve similar accuracy to the original network, with far fewer filters in CNN architecture. For example, a compressed network that uses half of the filters from the second layer of AlexNet to predict the spike rate of V4 neurons achieves comparable predictive accuracy to the full model. We further show that the DeepTune images from compressed images are consistent with the ones from the original uncompressed network. This is another form of stability in model interpretation.

Furthermore, we introduce RAR compression in chapter 4. RAR algorithm is similar to CAR, but the filters are removed based on their contribution to the final regression task. We employ RAR compression to prune the filters in each individual model for each neuron. Interestingly, when pruning 90% of the filters in the second layer of AlexNet, we observe a 5% increase in the average correlation coefficient between predicted and measured neural spike rates. The 10% remaining filters in the network have sufficiently diverse pattern selectivity to avoid any loss in V4 neuron models. Similar to CAR, the DeepTune visual for RAR is also stable and has patterns consistent with the original uncompressed DeepTune images.

**Future directions** The algorithms, methodologies and results presented in this thesis lead to several exciting future research directions. These directions are discussed in detail within each chapter. Here, we present a summary of these directions. In chapter 1, we discuss our unified framework for interpretable machine learning. However, investigating the theoretical aspects of interpretability and compression in machine learning models needs further study. A rigorous and unified technical definition of interpretability is an essential next step in this line of research. This definition should take into account both algorithmic interpretablity and post-hoc interpretation. Building theoretical frameworks to address connections between compression and interpretability is another direction which demands extensive study on the definition, derivation and evaluation of compressed interpretable artificial intelligence. Exploring the role of information theory and specifically channel coding in the compression of deep neural networks is one possible direction. Future work includes bringing together ideas from information theory, non-convex optimization, and manifold learning to better understand the error surface of neural networks and building a compressed interpretable DNN.

Alongside this theoretical avenue, further study is necessary to pursue the vital interactions between machine learning and computational neuroscience. In recent years, techniques, including two-photon calcium imaging and optogenetics, have emerged to simultaneously record and manipulate neural activity. In particular, calcium imaging has made it possible to record tens of thousands of neurons with fine-grained spatial and ad-

equate temporal resolutions. With the increasing interest in these imaging techniques, a more careful study on the interpretable and accurate models of neurons based on these modalities is necessary. Such progress could shed light on the problem of building state-of-the-art decoders of visual experience. With the recent progress in machine learning tools that encode brain activity, it becomes necessary to study their application in decoding human visual experience. When reconstructing the natural visual experience, one issue is that the natural scenes manifold is unknown. Models such as generative adversarial networks (GANs) [21] have shown remarkable capabilities to estimate the prior distribution of natural scenes and therefore, have application in brain decoders.

In many data-driven problems, elegant and transparent models such as scattering transform and random forests are not as accurate as deep neural nets, however, intelligently combining these models balances between the elegance and accuracy of final models. Such combined models are more interpretable and have invaluable implications in applications such as neuroscience. Additionally, a combined model sheds light on the limitations of each individual sub-model. For example, scattering transform presents a well-defined and transparent multi-layer model for vision. However, the predictive performance of models that are built upon this transform is considerably less than that of CNNs. This is primarily because, unlike scattering transform, CNNs are data-driven and have filters with more diverse functionalities. In a combined model of pre-defined scattering channels and learned CNN filters, visualizing CNN filters elucidates the missing components in scattering transform. A complete framework to build such models demands extensive theoretical and applied study and should be pursued in future.

# Chapter 2

# Structural compression of convolutional neural networks

## 2.1 Introduction

Deep convolutional neural networks (CNNs) achieve state-of-the-art performance for a wide variety of tasks in computer vision, such as image classification and segmentation [33, 40]. Recent studies have also shown that representations extracted from these networks can shed light on new tasks through transfer learning [48]. The superior performance of CNNs for large training datasets has led to their ubiquity in many industrial applications. Thus, CNNs are widely employed in many data-driven platforms such as cellphones, smart watches and robots. However, limited memory and computational power in these devices, along with the huge number of weights in CNNs, make necessary effective compression schemes.

There have been many works on compressing deep CNNs. These studies mostly focus on reducing the number and size of the weights or parameters by pruning and quantizing them. We call such compression schemes "weight compression". Optimal brain damage [36], optimal brain surgeon [24], Deep Compression [23] and most recently SqueezeNet [28] are some examples.

In addition to significant resource savings, compressed networks with fewer numbers of weights are easier to be investigated or interpreted by humans for possible domain knowledge gain. Here, interpretablity is loosely defined as the ability to explain or to present the decisions made by the model in understandable terms to a human [18]. Interpretability is typically studied from one of two perspectives. The first is algorithmic interpretablity and transparency of the learning mechanism. The other is post-hoc interpretability and explanation of the learned model using tools such as visualization. The first perspective

attempts to answer the question that how the model learns and works, while the second perspective describes the predictions without explaining the learning mechanism. Huge number of parameters in deep CNNs often cause difficulties in interpreting these network. From the perspective of post-hoc interpretability, a network with less number of parameters is easier to visualize and explain to human interpreter. Therefore, compression has a natural role in building more interpretable models. CNNs are often visualized using graphical explanation of their filters [74]. Thus to increase the interpretability, a compression method should decrease the number of filters. This is different from classical approach in compression of CNNs where individual weights are pruned and quantized [23].

For example, considering a convolutional neural network (CNN), convolutional filters are the smallest meaningful components of a CNN. Therefore, to uncover redundant information in a network and build more interpretable models, it is natural to compress CNNs based on removing "less important" filters. We call such schemes "structural compression" schemes. The challenge is in defining "filter importance". He et al. [26] and Li et al. [37] have studied structural compression based on removing filters and introduced importance indices based on average of incoming or outgoing weights to a filter. However, these importance measures typically do not yield satisfactory compressions of CNNs [26] because of the substantially reduced classification accuracy as a result. For example, classification accuracy for AlexNet decreases by 43% when we prune half of the filters in the first layer based on average incoming or outgoing weights.

Pruning activations or feature-maps to achieve faster CNNs has been also studied in [44]. Pruning activations can be viewed as removing filters in specific locations of the input, however, those filters almost always remain in other locations. Thus it seldom results in any compression of weights. On the other hand, pruning filters from the structure is equal to removing them for all the possible locations and avoiding storing them. Additionally, because of the simplified structure, filter-pruned networks are more interpretable compared to activation-pruned ones.

Pruning a fully-trained neural network has a number of advantages over training the network from scratch with less filters. A difficulty in training a network from scratch is not knowing which architecture or how many filters to start with and the huge numbers of possible architectures and filters would lead to high computational cost in a combinatorial manner as in other model selection problems [53]. Pruning provides a systematic approach to find the minimum number of filters in each layer required for accurate training. Furthermore, recent results suggest that for large-scale CNNs, the accuracy of the pruned network is slightly higher compared to a network trained from scratch ( [38] for VGG and ResNet, [30] for AlexNet). For small-scale CNNs, it is possible to train a network from scratch that achieves the same accuracy as the pruned network even though

the aforementioned computational cost is not trivial in this case. Additionally, in the majority of transfer learning applications based on well-trained CNNs, pruning algorithms achieve higher accuracies compared to training from scratch given the same architecture and number of filters [6, 44]. For example, [6] showed that a pruned AlexNet gains 47% more classification accuracy in bird species categorization compared to training the network from scratch.

Our main contributions in this chapter are three folds. First, we introduce a greedy structural compression scheme to prune filters in CNNs. A filter importance index is defined to be the classification accuracy reduction (CAR) (similarly for regression or RAR) of the network after pruning that filter. We then iteratively prune filters in a greedy fashion based on the CAR importance index. Our CAR structural compression scheme is shown to achieve much higher classification accuracies compared to other existing structural compression methods. CAR compressed AlexNet without retraining can achieve a compression ratio of 1.17 (for layer 1) to 1.5 (for layer 5) while having a close-to-original classification accuracy (or 54%). This is 21% to 43% higher than the compression ratio from the best benchmark method. If we fine-tune or retrain the CAR-compressed network, the compression ratio can be as high as 1.79 (for layer 3) when maintaining the same 54% classification accuracy. Second, we take advantage of weight pruning, quantization and coding by combining our method with Deep Compression [23] and report considerably improved compression ratio. For AlexNet, we reduce the size of individual convolutional layers by factor of 8 (for layer 1) to 21 (for layer 3), while achieving close to original classification accuracy (or 54% compared to 57%) through retraining the network. Third, we demonstrate the ability of our CAR algorithm to remove functionally redundant filters such as color filters making the compressed CNNs more accessible to human interpreters without much classification accuracy loss. To our knowledge, such a connection has not been reported previously. Furthermore, we introduce a variant of our CAR importance index that quantifies the importance of each image class to each CNN filter. A meaningful interpretation of each filter can be learned from the most and the least important class labels. This interpretation is consistent with the visualized pattern selectivity of that filter.

The rest of the chapter is organized as follows. In section 2, we introduce our CAR compression algorithm. The performance of the compression for the state-of-the-art CNNs in handwritten digit image and naturalistic image classification tasks is investigated in section 3. In section 4, we connect compression to the interpretation of CNNs by visualizing functionality of pruned and kept filters in a CNN. In section 5, a class-based interpretation of CNN filters using a variant of our CAR importance index is presented. The chapter is concluded in section 5.

## 2.2 CAR-based structural compression

### Notation

We first introduce notations. Let $w_i^L$ denote the $i^{th}$ convolutional filter in layer $L$ of the network and $n_L$ the number of filters in this layer ($i \in \{1, .., n_L\}$). Each convolutional filter is a 3-dimensional tensor with the size of $n_{L-1} \times f_L \times f_L$ where $f_L \times f_L$ is the size of spatial receptive field of the filter.

The activation or the feature map of filter $i$ in layer $L$ ($i = 1, .., n_L$) is:

$$\alpha_i^L = f(w_i^L * \mathcal{P})$$

where $f(\cdot)$ is the nonlinear function in convolutional network (e.g. sigmoid or ReLU) and $\mathcal{P}$ denotes a block of activations from layer $L-1$ (i.e. the input to the neurons in layer $L$). The activation for the first layer could be patches of input images to the convolutional network.

Assuming network $\mathcal{N}$ is trained on classification task, top-1 classification accuracy of network $\mathcal{N}$ is defined as:

$$Acc(\mathcal{N}) = \frac{N_{Correct}}{N_{Correct} + N_{Incorrect}}$$

where $N_{Correct}$ and $N_{Incorrect}$ are the number of correct and incorrect predicted classes, respectively.

In this chapter, we use FLOPs to quantify the computational cost in each convolutional layer of the neural network. FLOPs for each layer of the network equals to the number of floating-point operations required in that layer to classify one image. Let's assume $A \in \mathbb{R}^{n_{L-1} \times k_{L-1} \times k_{L-1}}$ is the input feature map and $B \in \mathbb{R}^{n_L \times k_L \times k_L}$ is the output feature map in layer $L$ where $k_L \times k_L$ is the spatial size. The FLOPs for this convolutional layer equals to $k_L^2 n_L f_L^2 n_{L-1}$. Additionally, the storage overhead for each convolutional layer of the network equals to $4 f_L^2 n_{L-1} n_L$ bytes [68].

### The proposed algorithm based on CAR importance index

In this section, we introduce our greedy algorithm to prune filters in layers of a CNN and structurally compress it. Figure 2.1 shows the process of greedy filter pruning. In each iteration, a candidate filter together with its connections to the next layer, gets removed from the network. The candidate filter should be selected based on an *importance* index of that filter. Therefore, defining an index of importance for a filter is necessary for any structural compression algorithm. Previous works used importance indices such as

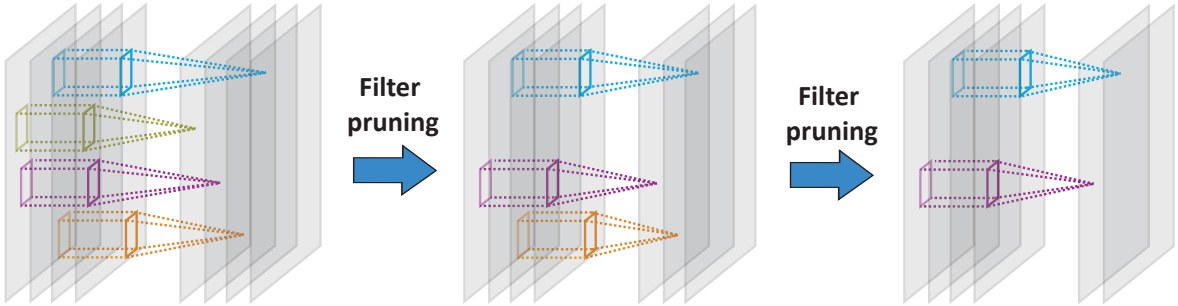


Figure 2.1: **Greedy compression of CNNs based on pruning filters.**

average of incoming and outgoing weights to and from a filter, but with unfortunately a considerable reduction of classification accuracy (e.g. 43% as mentioned earlier if one prunes only the first layer) for the compressed CNNs [26, 37]. To overcome this limitation, we define the *importance* measure for each filter in each layer as the classification accuracy reduction (CAR) when that filter is pruned from the network. That is,

$$CAR(i, L) = Acc(\mathcal{N}) - Acc(\mathcal{N}(-i, L))$$

where network $\mathcal{N}(-i, L)$ is network $\mathcal{N}$ except that filter $i$ from layer $L$ together with all of its connections to the next layer are removed from the network.

In our CAR structural (or filter pruning) compression algorithm, the filter with the least effect on the classification accuracy gets pruned in each iteration. The network can be retrained in each iteration and after pruning a filter. This process is regarded as *fine tuning* in this chapter. We present details of our fine tuning procedure in the next section. Algorithm 1 shows the pseudo code of our CAR greedy structural compression algorithm. Here, $n_{iter}$ and $r_{iter}$ are, respectively, the number of remaining filters and compression ratio in the current iteration.

One can also compress based on variants of our algorithm. One possibility is to avoid greedy process and remove several filters with lowest importance indices in one pass. This compression is faster, but the performance of the compressed network is worse than Algorithm 1 in the examples we tried. The greedy process with fine-tuning at each iteration seems to allow for a better data and network adaptation and improves compression performance.That is, in each iteration, we limit the algorithm to only find the least important filter prune it. In the next iteration, we update all the importance indexes using the new structure. This allows the algorithm to adapt to the new structure gradually and improves the classification accuracy.

**Algorithm 1** Greedy compression of CNNs based on pruning filters

**Input:** Weights in CNN, target layer $L$ with $n_L$ filters, target compression ratio $r_{target}$

Set $n_{iter} = n_L$ and $r_{iter} = 1$
**while** $r_{iter} < r_{target}$ **do**
  **for** $i = 1$ **to** $n_L$ **do**
    Compute $CAR(i, L)$, importance index for filter $i$ in layer $L$
  **end for**
  Remove the least important filter, $\arg\min_i CAR(i, L)$
  Update compression rate, $r_{iter} = n_L / n_{iter}$
**end while**

## 2.3 Compression rates and classification accuracies of the CAR compressed networks

To evaluate our proposed CAR structural compression algorithm, we have compressed LeNet [35] (with 2 convolutional layers and 20 filters in the first layer), AlexNet [33] (with 5 convolutional layers and 96 filters in the first layer) and ResNet-50 [25] (with 50 convolutional layers and 96 filters in the first layer). LeNet is a commonly used CNN trained for classification task on MNIST [35] consisting of 60,000 handwritten digit images. AlexNet and ResNet-50 are trained on the subset of ImageNet dataset used in ILSVRC 2012 competition [58] consisting of more than 1 million natural images in 1000 classes.

We used Caffe [29] to implement our compression algorithm for CNNs and fine tune them. The pre-trained LeNet and AlexNet are obtained from Caffe model zoo. All computations were performed on an NVIDIA Tesla K80 GPU. The CAR index is computed using half of the ImageNet test set. To avoid overfitting, the final performance of CAR compressed network is evaluated on the other half of the ImageNet test set. The running time of each pruning iteration depends on the number of filters remaining in the layer. On average, each iteration of CAR takes 45 minutes for the first layer of AlexNet. For the fine-tuning, the learning rate has been set to 0 for the layer that is being compressed, 0.001 for the subsequent layer and 0.0001 for all other layers. The subsequent layer is directly affected by the compressed layer, therefore, requires higher learning rate. The network is retrained for 500 iterations. This is sufficent for the classification accuracy to converge to the final value.

## LeNet on MNIST dataset

LeNet-5 is a four-layer CNN consisting of two convolutional layers and two fully-connected layers. CAR-compression has been performed on the convolutional layers and the performance on a hold-out test set is reported in Figure 2.2. We obtained classification accuracies (top-1) of the CAR-compression results (purple curve) and those from retraining or fine-tuning after CAR-compression on the same classification task (blue curve).

To compare the performance of our compression algorithm to benchmark filter pruning schemes, we have also implemented the compression algorithm based on pruning incoming and outgoing weights proposed in [26] and reported the classification accuracy curve in Figure 2.2. Furthermore, classification accuracy for random pruning of filters in LeNet has been shown in this figure. Candidate filters to prune are selected uniformly at random in this case. The error bar shows the standard deviation over 10 repeats of this random selection.

We conclude that our CAR-algorithm gives a similar classification accuracy to [26] for LeNet (using the outgoing weights in the first layer, and either weights for the second layer). Their accuracies are similar to the accuracy of the uncompressed, unless we keep very few filters for either layer. Fine-tuning improves the classification accuracy but there is not a considerable gap among performances (unless we keep very few filters, less than 8 among 20 for the first layer or less than 10 among 50 for the second layer). Among the 8 kept filters in the first layer, 4 of them are shared between the CAR-algorithm and that based on averaging outgoing weights in [26], while among the 10 kept filters in the second layer, 6 of them are shared.

## AlexNet on ImageNet dataset

AlexNet consists of 5 convolutional layers and 3 fully-connected layers. Figure 2.3 shows the classification accuracy of AlexNet on a hold-out test set after each individual convolutional layer is compressed using our proposed CAR algorithms or benchmark compression schemes.

Comparing the accuracies of compressed networks in Figure 2.3, there are considerable gaps between our proposed CAR-algorithm (purple curves) and the competing structural compression schemes that prune filters [26] for all five layers. Further considerable improvements are achieved by retraining or fine-tuning the CAR-compressed networks (see the blue curves in Figure 2.3).

Pruning half of the filters in either of the individual convolutional layers in AlexNet, our CAR algorithm achieves 16% (for the layer 5) to 25% (for the layer 2) higher classification

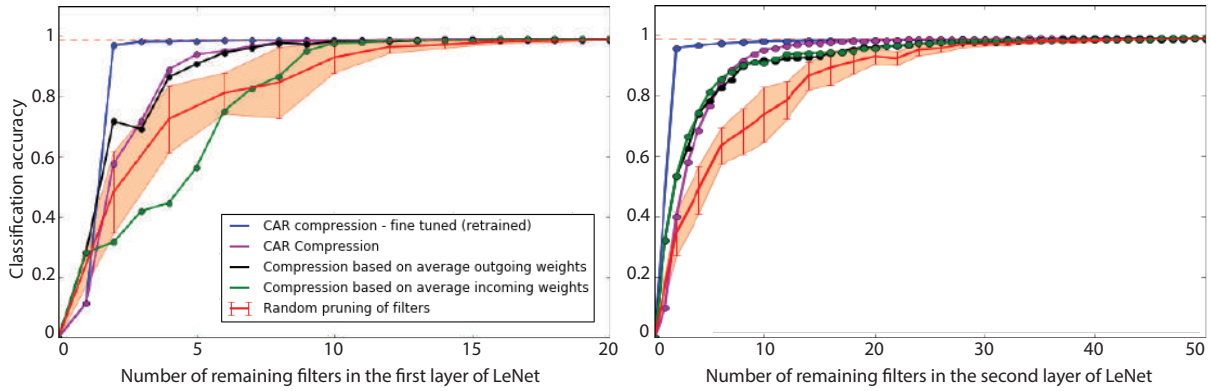


Figure 2.2: **Performance of CAR compression for LeNet.** The top figure shows the overall classification accuracy of LeNet when the first convolutional layer is compressed. The bottom figure shows the classification accuracy when the second convolutional layer is compressed. The classification accuracy of uncompressed network is shown with a dashed red line. The purple curve shows the classification accuracy of our proposed CAR compression algorithm for various compression ratios. The accuracy for the fine tuned (retrained) CAR compression is shown in blue. The black and green curves shows the accuracy for compressed network based on outgoing and incoming weights, respectively. The red curve shows the accuracy when filters are pruned uniformly at random. The error bas is reported over 10 repeats of this random pruning process.

accuracies compared to the best benchmark filter pruning scheme (pruning based on average outgoing weights). If we retrain the pruned network, it achieves 50% to 52% classification accuracy (compared with 57% of the uncompressed AlexNet). The superior performance of our algorithm for AlexNet is due to the proposed importance index for the filters in CNN. This figure demonstrates that our algorithm is able to successfully identify the least important filters for the purpose of classification accuracy. In section 5.2, we discuss the ability of our compression scheme to reduce functional redundancy in the structure of CNNs.

To present a different but equivalent quantitative comparison, we have reported the compression ratio and feed-forward speed up in Table 2.1. Each individual convolutional filter is pruned while the classification accuracy dropped a relative 5% from the accuracy of uncompressed network (i.e. 54% compared to 57%). Results for CAR compression with and without fine tuning and compression based on average incoming and outgoing weights are presented in this table. The CAR algorithm (without retraining) can achieve a compression ratio of 1.17 (for layer 1) to 1.50 (for layer 5), which is 21% to 43% higher

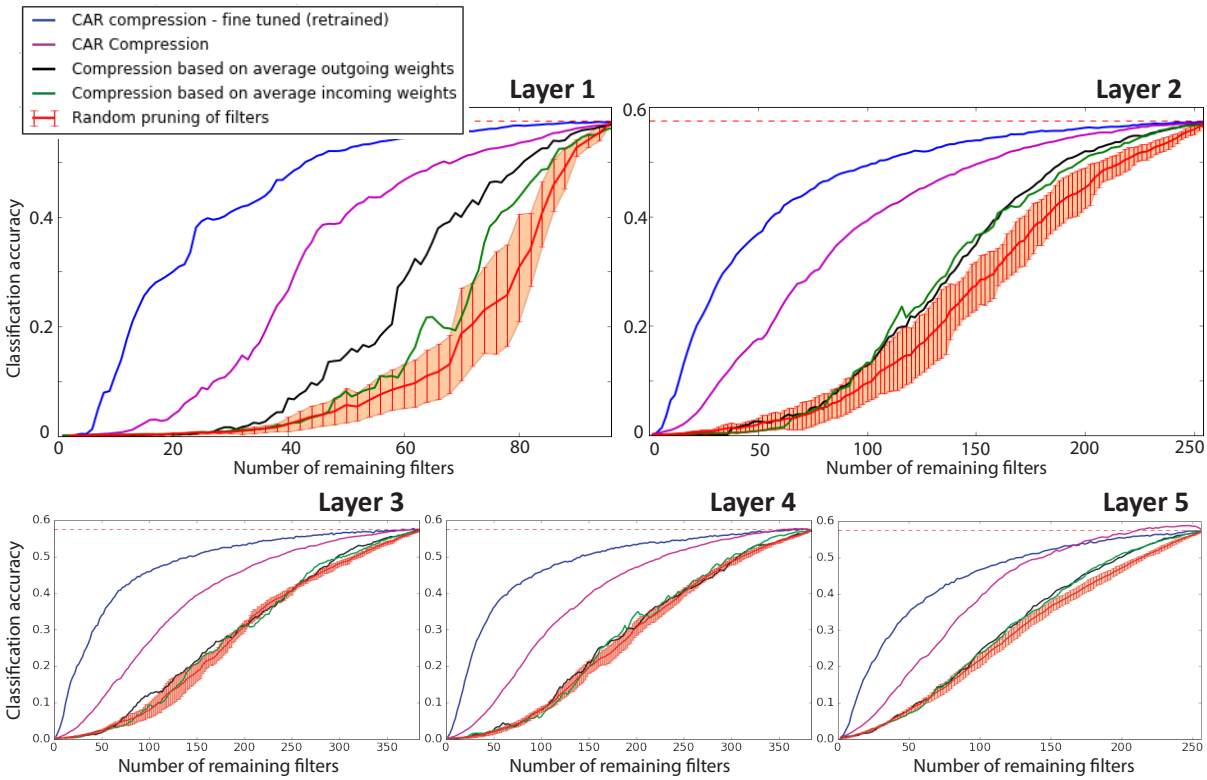


Figure 2.3: **Performance of CAR compression for AlexNet.** Each panel shows the classification accuracy of the AlexNet when an individual convolutional layer is compressed. In each panel, the classification accuracy of uncompressed network is shown with a dashed red line. The purple curve shows the classification accuracy of our proposed CAR compression algorithm for various compression ratios. The accuracy for the fine tuned (retrained) CAR compression is shown in blue. The black and green curves shows the accuracy for compressed network based on outgoing and incoming weights, respectively. The red curve shows the accuracy when filters are pruned uniformly at random. The error bas is reported over 10 repeats of this random pruning process.

than those from the benchmark methods. If we fine-tune or retrain the CAR-compressed network, the compression ratio can be as high as 1.79 (for layer 3) when maintaining the same 54% classification accuracy.

Table 2.1: **Comparison of compression performance between our greedy CAR compression algorithm and benchmark schemes on AlexNet.** For the compressed networks, the filters are pruned while the classification accuracy dropped a relative 5% from the accuracy of original network (i.e. 54% compared to 57%). FLOPs equals to the number of floating-point operations required in each layer to classify one image.

| Layer | Compression method | Number of remaining filters | Bytes | FLOPs | Compression ratio & Feed-forward speed up |
|---|---|---|---|---|---|
| Layer 1 | Original | 96 | 0.14M | 105.41M | - |
| | Incoming weights | 90 | 0.13M | 98.82M | 1.07× |
| | Outgoing weights | 88 | 0.13M | 96.63M | 1.09× |
| | CAR | **82** | **0.12M** | **90.04M** | **1.17**× |
| Layer 2 | Original | 256 | 1.23M | 223.95M | - |
| | Incoming weights | 223 | 1.07M | 195.08M | 1.15× |
| | Outgoing weights | 217 | 1.04M | 189.83M | 1.18× |
| | CAR | **189** | **0.91M** | **165.33M** | **1.35**× |
| Layer 3 | Original | 384 | 3.54M | 149.52M | - |
| | Incoming weights | 342 | 3.15M | 133.17M | 1.12× |
| | Outgoing weights | 334 | 3.08M | 130.05M | 1.15× |
| | CAR | **287** | **2.64M** | **111.75M** | **1.34**× |
| Layer 4 | Original | 384 | 2.65M | 112.14M | - |
| | Incoming weights | 332 | 2.29M | 96.95M | 1.16× |
| | Outgoing weights | 346 | 2.40M | 101.04M | 1.11× |
| | CAR | **279** | **1.93M** | **81.48M** | **1.38**× |
| Layer 5 | Original | 256 | 1.77M | 74.76M | - |
| | Incoming weights | 220 | 1.52M | 64.25M | 1.16× |
| | Outgoing weights | 222 | 1.53M | 64.83M | 1.15× |
| | CAR | **171** | **1.18M** | **49.94M** | **1.50**× |
| Layer 1 | Fine-tuned CAR | 58 | 0.08M | 63.69M | 1.66× |
| Layer 2 | | 153 | 0.73M | 133.84M | 1.67× |
| Layer 3 | | 214 | 1.97M | 83.33M | 1.79× |
| Layer 4 | | 225 | 1.56M | 65.71M | 1.71× |
| Layer 5 | | 176 | 1.22M | 51.40M | 1.45× |

## Combination with Deep Compression

One advantage of our CAR-algorithm is that it is amenable to combination with weight based compression schemes to achieve substantial reduction in memory usage.

Table 2.2: **Compression performance of CAR-algorithm combined with Deep Compression**

| Layer | Weight pruning + Quantization [23], $Acc = 0.57$ | Weight pruning + Quantization + Huffman Coding [23], $Acc = 0.57$ | CAR + Weight pruning + Quantization, $Acc = 0.54$ | CAR + Weight pruning + Quantization + Huffman Coding, $Acc = 0.54$ |
|---|---|---|---|---|
| Layer 1 | 3.07× | 4.87× | 5.13× | **8.13**× |
| Layer 2 | 6.90× | 10.60× | 11.52× | **17.70**× |
| Layer 3 | 7.63× | 11.85× | 13.66× | **21.21**× |
| Layer 4 | 7.09× | 10.98× | 12.13× | **18.77**× |
| Layer 5 | 7.14× | 10.60× | 10.36× | **15.38**× |

Deep Compression [23] is a recent weight-based compression procedure that uses weight pruning, quantization and huffman coding. We have performed Deep Compression on top of our proposed compression algorithm and reported the compression ratio for AlexNet in Table 2.2. Again, the filters are pruned while the classification accuracy is in the range of relative 5% from the accuracy of uncompressed network (54% compared to 57%). An additional 5 fold (for layer 1) to 12 fold (for layer 3) increase in compression ratio is acheived through joint CAR and Deep Compression. That is, further weight compression boosts the compression ratio by sparsifying weights of the kept filters, although the number of filters is the same as the CAR compression.

## ResNet-50 on ImageNet dataset

First introduced by He et al. [25], deep residual networks take advantage of a residual block in their architecture (Figure 2.4, right panel) to achieve higher classification accuracy compared to a simple convolutional network. We have studied the performance of CAR compression on ResNet-50 architecture [25] with 50 layers of convolutional weights. Figure 2.4 top left panel shows the classification accuracy of ResNet-50 after pruning first convolutional layer using CAR algorithm or benchmark compression schemes. The bottom panel shows the classification accuracy after pruning the first convolutional layer in the first residual block (layer *Conv a - Branch 2* in Figure 2.4). CAR pruning of other convolutional layers in this residual block or higher blocks yields to similar figures and

are not shown here to avoid redundancy. These accuracies are reported on the ILSVRC 2012 ImageNet hold out test set.

It is of great interest to compare at high compression ratio regimes where we keep less than 30 filters out of 64. In this situation and pruning layer *Conv 1*, the CAR algorithm (purple curve in Figure 2.4) outperforms the competitors based on incoming and outgoing weights. The higher the compression ratio, the higher the improvements by the CAR algorithm. For low compression ratio regimes, the performances are similar. Compared to AlexNet, the gap between CAR and benchmark compressions is smaller for the first layer. This might be an evidence that ResNet has less redundant filters. Retraining (fine-tuning) the CAR-compressed network achieves further improvement in classification accuracy (blue curve in Figure 2.4). In fact, our CAR-algorithm achieves 72% classification accuracy (compared with the 75% for the uncompressed ResNet-50) when pruning half of the filters in the first layer of ResNet-50. This accuracy is 15% higher than that of filter pruning based on average outgoing or incoming weights.

For the residual block, we have pruned layer *Conv a - Branch 2* and reported the classification accuracy in Figure 2.4. The accuracy of CAR algorithm is almost similar to the compression based on incoming and outgoing weights. Interestingly, the accuracy drops less than 15% if we fully prune the filters in this layer i.e. remove branch 2 from the residual block. The drop in accuracy is less than 5% for the fine-tuned network. The main reason for this is the existence of shortcuts in the residual module. The uncompressed branch 1 that is a parallel channel with the pruned filter allows the information to transfer through the residual layer.As a result of these parallel channels in the residual blocks, deep residual networks are more robust to pruning filters compared to simple convolutional networks.

## 2.4 CAR-compression algorithm prunes visually redundant filters

To study the ability of our proposed CAR compression method to identify redundant filters in CNNs, we take a closer look at the compressed networks. In this section, we focus on the second layer of AlexNet. It has 256 filters with visually diverse functionalities, which is considerably more than the numbers of filters in LeNet or the first layer of AlexNet. It is also easier to visualize filters in this layer compared to higher layers of AlexNet. However, similar results hold for the other layers of AlexNet and LeNet. Recall that we first performed CAR structural compression to prune 256 filters in the second layer of AlexNet, and continued to iterate the algorithm while the classification accuracy is 54% or within a relative 5% from the accuracy of uncompressed network. This led to

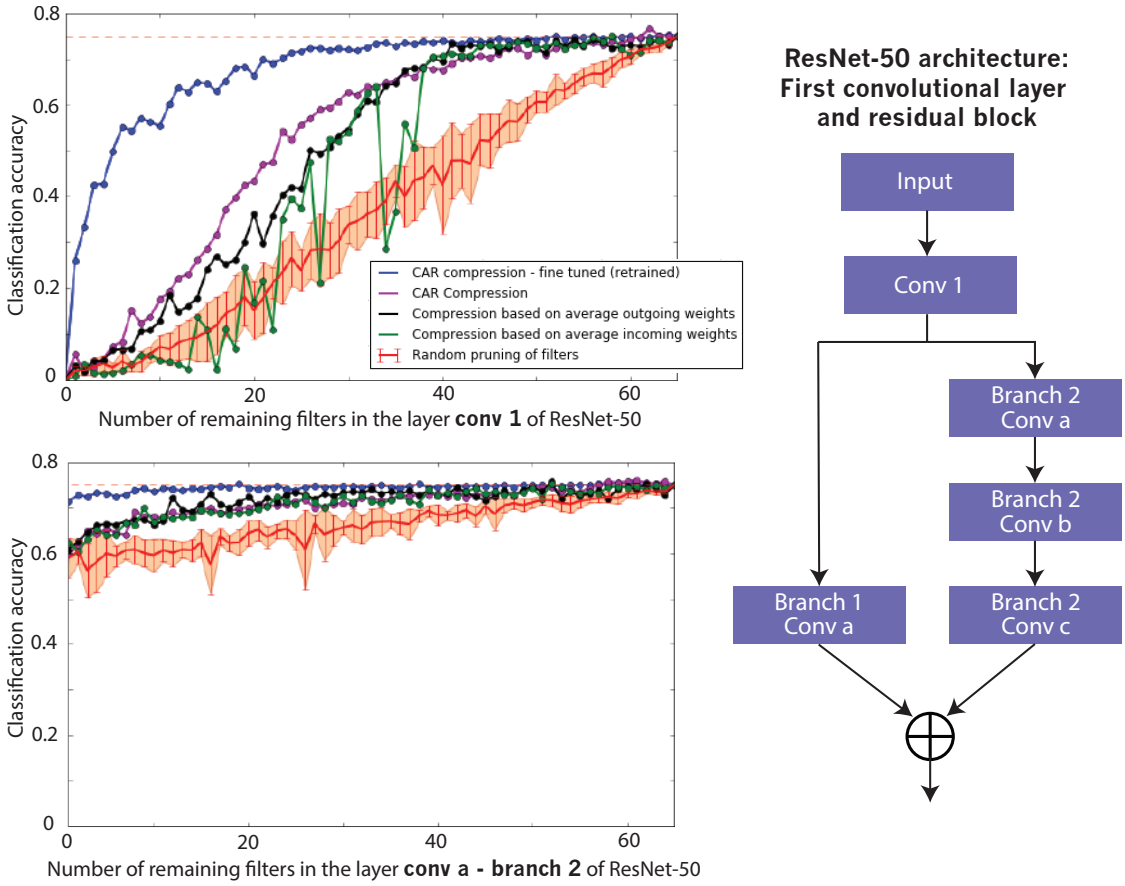


Figure 2.4: **Performance of CAR compression for ResNet-50.** The top left figure shows the classification accuracy of the ResNet-50 when the first convolutional layer is compressed. The bottom left figure shows the classification accuracy when filters in the first residual module layers are pruned (with the first layer untouched). The classification accuracy of uncompressed network is shown with a dashed red line. The purple curve shows the classification accuracy of our proposed CAR compression algorithm for various compression ratios. The accuracy for the fine tuned (retrained) CAR compression is shown in blue. The black and green curves shows the accuracy for compressed network based on outgoing and incoming weights, respectively. The red curve shows the accuracy when filters are pruned uniformly at random. The error bas is reported over 10 repeats of this random pruning process. The right panel shows the architecture of first layers in ResNet-50.

pruning 103 filters out of 256 filters in this layer. A subset of the removed and remaining filters are visualized in Figure 2.5. To visualize the pattern selectivity of each filter, we have fed one million image patch to the network and showed the top 9 image patch that activate a filter. This approach has been previously used to study functionality of filters in deep CNNs [74]. We have manually grouped filters with visually similar pattern selectivity (blue boxes in 2.5). A hand-crafted image has been shown beside each group to demonstrate the function of filters in that group. Our algorithm tends to keep at least one filter from each group, suggesting that our greedy filter pruning process is able to identify redundant filters. This indicates that pruned filters based on the CAR importance index have in fact redundant functionality in the network.

Looking deeper into the CAR indexes, out of top 20 pruned filters, 17 of them in the first layer and 14 of them in the second layer correspond to the color filters, respectively. This finding points to the fact that shape is often first-order important for object recognition. To further investigate the effect of compression of each of the convolutional layers, we have shown the scatter plots of the classification accuracy for each of the 1000 classes in ImageNet in Figure 2.6. Although the total classification accuracy is about a relative 5% lower for the each compressed network, the accuracies for many of the categories are comparable between compressed and uncompressed networks. In fact, 37% (for layer 5) to 49% (for layer 2) of the categories have accuracies no larger than 3% below those for the uncompressed network.

## 2.5 Class-based interpretation of filters

With a slight modification in our definition for the CAR importance index, we build a new index that enables us to interpret the filters in CNNs with respect to image classes. We define $CAR^c(i, L)$ to be classification accuracy reduction in class $c$ when filter $i$ in layer $L$ is pruned. $CAR^c$ identifies the set of classes that their classification accuracy highly depends on the existence of a filter. These classes are the ones with highest $CAR^c$ among all of the classes. Similarly, for each filter, the performance in classes with the smallest $CAR^c$ have less dependency to that filter. Note that both CAR and $CAR^c$ indexes could be negative numbers, that is the pruned network has higher classification accuracy compared to the original network. The labels of the two sets of classes with highest and lowest $CAR^c$ present a verbal interpretation of each filter in the network. $CAR^c$-based interpretation is a better fit for the higher layers in the CNN because filters in these layers are more abstract and therefore more explainable by the class labels. In Figure 2.7 we show that the interpretation based on $CAR^c$ is consistent with the visualized pattern selectivity of each filter in layer 5 of AlexNet. Similar to previous section, the

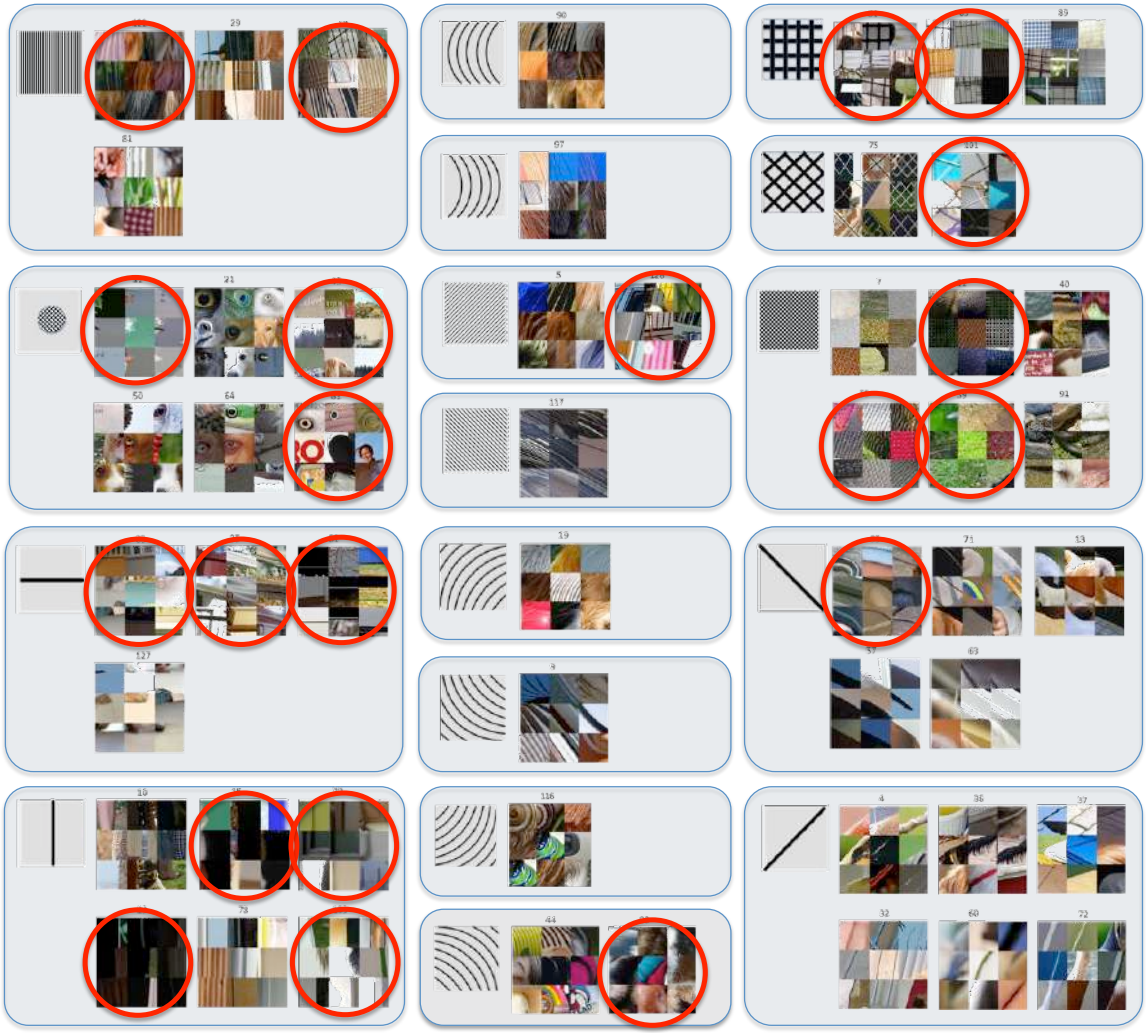

Figure 2.5: **CAR compression removes filters with visually redundant functionality from second layer of AlexNet.** To visualize each filter, we have fed one million image patch to the network and visualized each filter by 9 image patches with top response for that filter. We have manually clustered 256 filters in the second layer of AlexNet into 30 groups (17 of them visualized here). Pattern selectivity of each group is illustrated in the left top corner of each box using a manually designed patch. We continue to iterate the CAR-based algorithm while the classification accuracy is in the relative range of 5% from the accuracy of uncompressed network. This leads to pruning 103 out of 256 filters in this layer. The pruned filters are specified with a red circle.

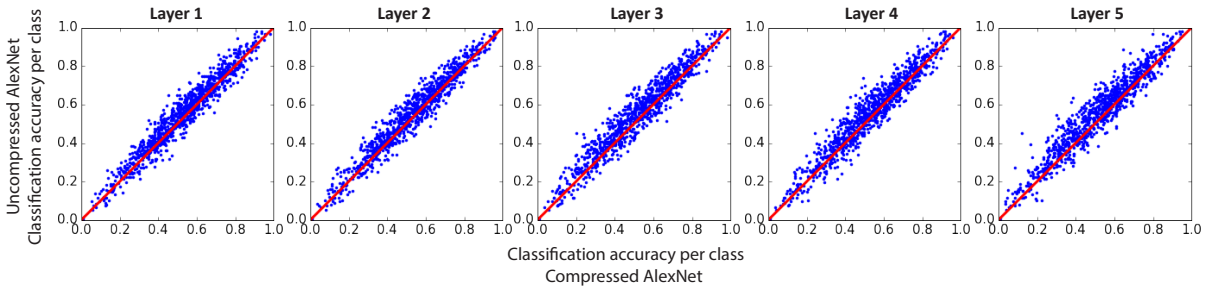


Figure 2.6: **Comparison of classification accuracy between compressed and uncompressed networks for each class of image in AlexNet.** Each panel corresponds compression in one layer. Each point in plots corresponds to one of the 1000 categories of images in ImageNet test set.

visualization is based on top 9 image patch activating that filter. We have selected three filters in layer 5 that are among the most important filters in this layer based on our original CAR pruning. Similar to the visualization in Figure 2.6, panel A illustrates the top 9 image patches that activate each filter. Panels B and C show the top and bottom 5 classes with highest and lowest $CAR^c$, respectively. Besides the class label, one sample image from that class is also visualized. Some of these classes are pointed out with green arrows in the scatter plot of classification accuracy for 1000 classes in ImageNet (panel D). The classes with highest $CAR^c$ share similar patterns visible in the top 9 patch activating each filter. For filter 1, the smooth elliptic curvature that consistently appears in the classes such as *steep arch bridge* or *soup bowel* is visible in the top activating patches (Panel A). On the other hand, less elliptic curvature patterns are expected in classes such as *mailbag* or *altar*. Filter 2 has higher $CAR^c$ for classes that contains patterns such as insect or bird's head. Filter 3 is mostly selected by the classes that contain images of a single long tool, particularly musical instruments such as *oboe* or *banjo*.

## 2.6 Discussion

Structural compression (or filter pruning) of CNNs has the dual purposes of saving memory cost and computational cost on small devices, and of resulted CNNs being more humanly interpretable. In this chapter, we proposed a greedy filter pruning based on the importance index of classification accuracy reduction (CAR). We have shown with AlexNet that the huge gain (8 to 21 folds) in compression ratio of CAR+Deep Compression schemes, without a serious loss of classification accuracy. Furthermore, we saw that the

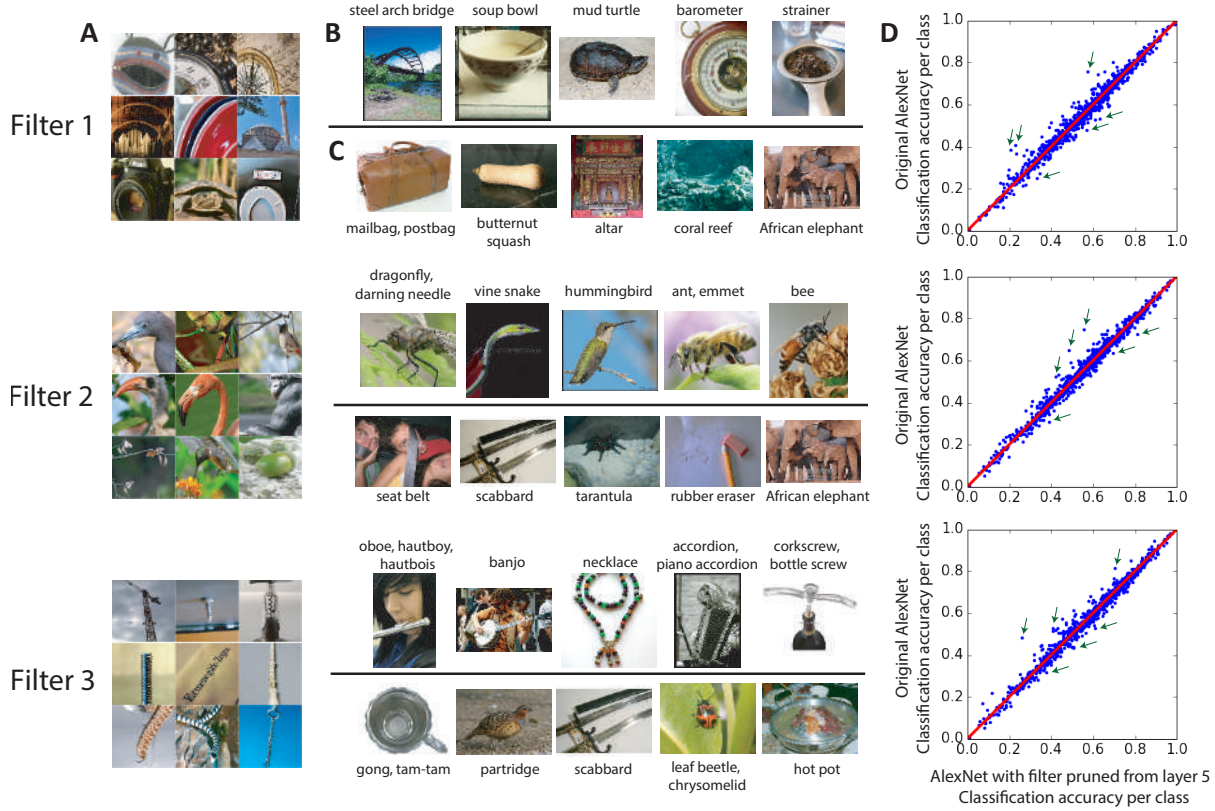


Figure 2.7: **The interpretation based on $CAR^c$ is consistent with the visualized pattern selectivity of each filter in layer 5 of AlexNet.** Panel A shows The top 9 image patches that activate each filter [74]. Panel B and C show the top and bottom 5 classes with highest and lowest $CAR^c$, respectively. Besides the class label, one sample image from that class is also visualized. Panel D shows the scatter plot of classification accuracy for each of the 1000 classes in ImageNet. Three of the top and bottom classes with highest and lowest $CAR^c$ are pointed out with green arrows. Each row corresponds to one filter in layer 5 of AlexNet.

pruned filters have redundant functionality for the AlexNet. In particular, for many categories in ImageNet, we found that the redundant filters are color-based instead of shape-based. This suggests the first order importance of shape for such categories.

However, a greedy algorithm is likely to be sub-optimal in identifying the best candidate filters to drop. The optimal solution may be to search through all possible subsets of filters to prune, but this can be computationally expensive and may lead to over-pruning. Procedures for subset selection, including genetic algorithms and particle swarm opti-

mization, could be helpful in the compression of CNNs and will be investigated in future work. Even though the CAR compression of ResNet achieves state-of-the-art classification accuracy among other structural compressions by pruning the identity branch and identifying the redundant connections. ResNet compression merits further investigation because of the identity branches in the residual blocks.

We also proposed a variant of CAR index to compare classification accuracies of original and pruned CNNs for each image class. In general, we can compare any two convolutional neural networks that are trained on the similar dataset through this index. The comparison could be done by looking into set of classes that are important for each filter in layer of each network. A similar class-based comparison for any two networks through our importance index is possible. This is a fruitful direction to pursue, particularly given the recent wave of various CNNs with different structures. Finally, we expect that our CAR structural compression algorithm for CNNs and related interpretations can be adapted to fully-connected networks with modifications.

# Chapter 3

# Computational models for V4 neurons

## 3.1 Introduction

Recent advancements of technologies have created tremendous potentials for understanding brain structure and function. These potentials manifest themselves mainly in two directions: the huge amounts of brain data collected in different modalities (e.g. electrophysiological single-cell recordings or imaging data) and the rapid machine learning algorithm developments that are able to harness not only dramatically increased computing resources but also large data using human intelligence through crowd sourcing. Deep learning algorithms are among the most notable such algorithms. These algorithms have been successful in many fields of artificial intelligence and machine learning such as computer vision, robotics and speech recognition. It is natural to question whether these algorithms suited for artificial intelligence systems could help our understanding of brain function.

Understanding the function of primate visual pathways has been a major problem in computational neuroscience. On the ventral visual pathway, cortical area V4 is a large retinotopically-organized area located intermediate between the early primary visual cortex areas such as V1 and V2 and high-level areas in the inferior temporal lobe. Although believed to be crucial for visual object recognition and visual attention, area V4's function remains mysterious. In particular, Area V4's highly nonlinear response properties [20] has been one of the major difficulties to construct quantitative models that accurately describe how visual information is represented in V4.

Area V4's function is believed to be much more complex than that in area V1. While it has been shown that the simple cells in the primary visual cortex area V1 can be fairly

accurately modeled using a few parameters such as spatial, frequency, and velocity tuning [27, 14, 8], these methods fail to generalize directly to area V4. This is mainly due to its highly nonlinear response [65] and diverse tuning properties [56]. There has been many studies concerning the tuning properties and shape representation of V4 neurons. However, the current knowledge about V4 is far from providing a unified view on concrete and consistent tuning properties of V4 neurons [56]. Gallant et al. [19, 20] have found that V4 neurons are most selective for non-Cartesian gratings containing multiple orientations by comparing V4's response Cartesian gratings and polar and hyperbolic (non-Cartesian) gratings stimulus. It has also been discovered by Pasupathy and Connor [51, 50] that V4 neurons are selective to curved contour features by using a parameterized set of contour stimulus varying in angularity, curvature, and orientation. More recently, studies based on spectral receptive model [12], which accounts for second-order nonlinear response properties, successfully enhance the understanding of V4's orientation tuning properties. However, the lack of good quantitative models with prediction accuracies similar to those for the V1 area and good visualization of V4's tuning properties suggest that V4 neurons might be more complex than those captured by the current quantitative models [56]. Therefore, a more precise characterization of shape selectivity with higher-order non-linear models is necessary for neurons in this V4 cortical area .

On the other hand, deep learning algorithms especially convolutional neural networks (CNN), trained on large-scale datasets such as ImageNet [15], have led to higher performance models than traditional methods in many artificial intelligence tasks such as object recognition. These CNNs not only are well suited for object recognition tasks [33], but also provided transferable features for a wide range of vision tasks [61] such as image segmentation and image captioning. Since the structures of these CNNs are inspired by the hierarchical structure of neural networks in human visual cortex, one would expect that these CNN features are also transferable and predictive of real neurons' activity in human or macaque's brains. Indeed, fueled by innovation in the computer vision and artificial intelligence communities, recent developments in computational neuroscience have been using CNNs to make strides in modeling neural single-unit and population responses in visual cortical areas [70]. The early use of artificial neural networks for modeling receptive field properties of simple and complex cells in area V1 dates back to Prenger et al. [52]. More recently, Yamins et al. [71] and Cadieu et al. [7] have investigated the similarity between the representational performance of deep convolutional neural networks and the V4/inferior temporal (IT) cortex on a image classification task. These results show that deep neural networks have the potential to be used to build quantitative predictive models of neural processing, however, it is not clear whether this phenomenon generalizes other vision tasks, and how to interpret these complex CNN models to gain scientific insights about the visual cortex.

While fitting deep learning models have been made computationally easier recently, interpreting and understanding the deep convolutional neural networks remains a difficult research topic [74, 72]. Due to its nonlinear properties and parameter sharing properties, it is much more difficult to associate an interpretation to the weight parameters in a deep convolutional neural network based model than those in a simple linear regression model. Another major concern of the neural network based modeling approach is that conclusions about neuronal behavior based on *in silico* models of neurons will be overwhelmingly dependent on the complicated details of the modeling procedure. We find many deep neural network architectures lead to similar prediction performance, Given high prediction performance, stability relative to the model and data perturbation is a minimum requirement for the interpretability and reproducibility of data driven results [73]. Therefore, only stable part of the results from the models should be interpreted.

In this work, we propose a deep convolutional neural network based predictive model for V4 neurons and a stability based interpretation and visualization framework for understanding these deep models to address aforementioned questions. Our analysis use input-response data of 71 well-separated V4 neurons in two behaving macaques collected at the Gallant Lab [67]. For each neuron, the inputs are randomized natural images from a black-white image database and the responses are spike response rates. We first build state-of-the-art predictive models (with prediction error as a measure of goodness of fit) for these 71 V4 neurons (that outperform V1 Gabor models and previous STRF models for V4 neurons). We then apply the stability principle [73] to seek neuron characterizations through stable tuning regions and stable smooth DeepTune patterns across 18 predictive models of similar state-of-the-art prediction accuracies. Specifically, instead of training end-to-end nonlinear models, we employ transfer learning [49, 61] to build accurate predictive models for V4 neurons with accuracies similar to those for the V1 neurons. In our framework, "transfer" happens in three different forms: we transfer CNNs to different black-white image data than they were trained on (color-images), and we transfer from macro classification tasks to a regression problem at the micro neuron level, and we transfer from human (label response) data to macaque (neuron response) data. Finally, we show that, by examining the stable part of our smooth DeepTune images, we can precisely characterize V4 neurons' tuning to curvature or texture patterns and explain the function of suppressive V4 neurons. Our characterization goes beyond the previous methods such as the spectral receptive field model and demonstrates V4 tunings for a wide variety of patterns in natural images such as curvatures and textures that are generated from natural input images (instead of synthetic images). Because smooth DeepTune images provide a natural-looking image, these results are testable as they could serve as input stimuli for future experiments.

## 3.2 CNN-based strongly predictive and stable models for individual V4 neurons

To study the ability of CNNs in explaining visual cortical area V4, we have recorded spike rates from 71 well isolated neurons in V4 from two awake-behaving male macaque. These recordings have been previously used to study the sparseness of neural codes in the area V4 [67]. The stimuli consisted of a random sample of circular patches of grayscale digital photographs from a commercial digital library (Corel). Random images were then concatenated into long sequences so that each 16.7 ms frame contained a random image patch from the library. All images were centered on the estimated classical receptive field (CRF, see *Appendix, Text*, Data Collection for CRF estimation procedure) and patch size was adjusted to be two to four times the CRF diameter (Figure 3.1-C). The training data set for each neuron consists of $4,000 - 12,000$ natural images. Spike count was measured at 60 Hz, resulting in two measurements per image. For the holdout test dataset, 300 images were shown for each neuron in a fixed order, distinct from the images shown for the training dataset. The sequence of test images was repeated; on average, each image in the test dataset was shown 9.3 times. The resulting spike counts were averaged to provide a lower-variance estimate of the expected spike count; repeats also allowed for estimation of the amount variance in the neuron explainable by the stimulus image (see *Appendix, Text*, Data Collection for additional details).

We propose the following transfer learning framework to analyze our V4 input-response data illustrated in Figure 3.1. For a given layer of a pre-trained convolutional neural network, in the first stage (Figure 3.1-A), we extract features as intermediate outputs of CNN for each stimulus image (represented to the CNNs as RGB color images by making all three color channels the same grayscale intensities); In the second stage (Figure 3.1-B), these features are served as predictors in a linear regression method with spike-rates as the response. Formally, for one stimulus image at time $t$ as $\mathbf{z}_t \in \mathbb{R}^{s\times s}$ ($s = 227$ in AlexNet model), the given layer of CNN transforms this image into a flattened feature vector $\mathbf{x}_t \in \mathbb{R}^d$. This feature transform is denoted as function $h : \mathbb{R}^{s\times s} \mapsto \mathbb{R}^d$. We then regress the spike count of the neuron at time $t$ as $y_t \in \mathbb{R}$ against the feature vector $\mathbf{x}_t$. V4 neuron's behavior is sensitive to the recent history of images shown to the subject; hence, in practice, to build an accurate predictive model of expected spike count, we regress $y_t$ against the image features from last $k$ frames of video prior to and including time $t$, i.e. $\mathbf{z}_t, ..., \mathbf{z}_{t-k+1}$. The time lag $k$ is set to be 9 (consisting frames at 0, 16.7, ..., 150.3 ms) as the common practice (e.g. in [66, 12]).

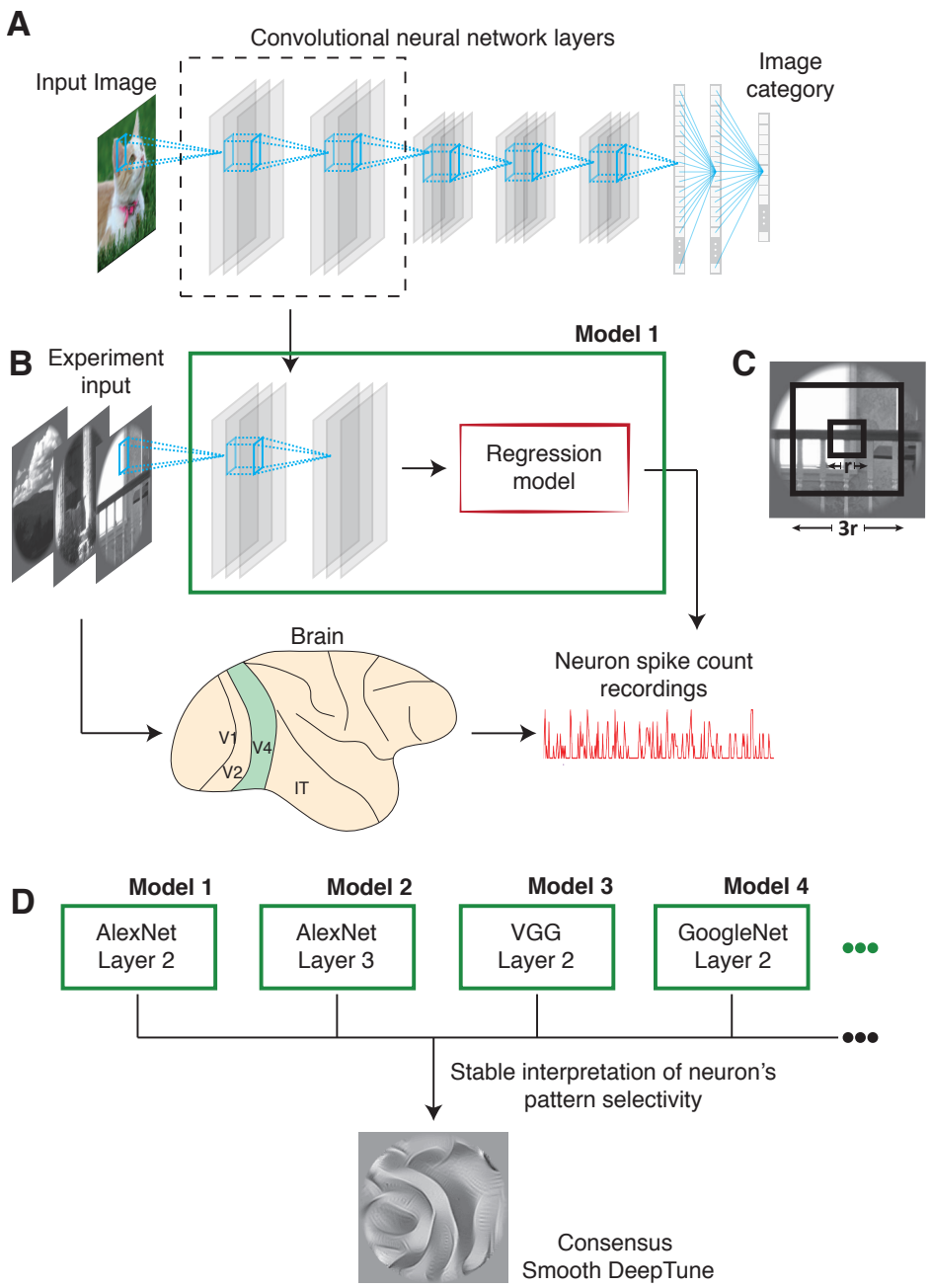


Figure 3.1: **Transfer learning of pre-trained convolutional neural networks to predict spike rates of neurons in the area V4 and the stability-driven interpretation. A.** A convolutional neural network trained to perform 1000-class image classification task on ImageNet dataset which contains more than 1.2 million natural images. In this chapter, three well-known CNN structures have been used: AlexNet, VGG and GoogleNet. **B.** Weights from early layers of the trained CNN are fixed and transfered

to extract features from stimulus images in animal experiments. Each stimulus image, represented as grayscale pixel intensities, is propagated forward to CNN, yielding a vector representation of the image. Then, a regularized linear regression model is constructed to predicts spike rates of each V4 neuron from this vector representation. The linear model is learned via the spike rates in training set and evaluated on a hold-out test set. The training set contains 4,000-12,000 natural images. The test set contains 300 images where spike rates are averaged across ten repeats. **C.** The classical receptive field (CRF) during the experiment is set in the middle of the stimuli with width $r$ while the whole image has the width $3r$. **D.** To discover the stable pattern selectivity of each V4 neruon, We fit 18 models using features from layers 2, 3, 4 of the three trained convolutional networks (AlexNet, GoogleNet, VGG), with either $\ell_1$ or $\ell_2$ regularization in linear model. Then "smooth DeepTune", a stability-driven CNN-based model visualization framework is used to characterize V4 neurons' tuning preferences. This visualization is built upon finding the optimal stimulus for each neuron from each model and interpret the stable pattern across all models (More details in the next section). The consensus smooth DeepTune visualization for this neuron among 18 models shows a stable curvature pattern with edges forming an approximately 90 degree angle.

---

Finally, our predictive model for a single neuron response takes the following form

$$F : \mathbb{R}^{s\times s\times k} \to \mathbb{R}$$
$$(\mathbf{z}_t, ..., \mathbf{z}_{t-k+1}) \mapsto \sum_{j=0}^{k-1} \boldsymbol{\beta}_{j+1}^T h(\mathbf{z}_{t-j}),$$

where $(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k) \in \mathbb{R}^{d\times k}$ are the regression parameters to be determined and $h$ is the fixed CNN feature transform.

The CNNs we used in this chapter are pre-trained CNNs for classification tasks on ImageNet dataset from the ImageNet Large Scale Visual Recognition Challenge [58]. In particular, we used three CNNs, AlexNet [33], GoogleNet [63] and VGG [62], with implementation details provided by the deep learning package Caffe [29]. Layers 2, 3, 4 of the three pre-trained convolutional networks are the main focus of our analysis (See *Appendix, Text*, Stability for the results on other layers). Because of the pooling operations, normalization operations, and non-linear activation functions in the CNNs, the CNN features are complex nonlinear functions of the raw input image. These CNN features are the outcome of learning from the large scale image dataset ImageNet, and are in general hard to explain via mathematical formula. To better understand the feature extracted via CNN, we provide a visualization of these features in *Appendix, Figs. S3, S4 and S5*.

We observe that these features encode more complex patterns than the standard Gabor wavelets do. This observation makes up our initial insight that CNNs could help model V4 neurons which represent more complex patterns than V1 neurons [56].

To perform the regression analysis, we solve the following regularized linear regression problem

$$\left(\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_k\right) = \underset{\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_k}{\arg\min} \frac{1}{2} \sum_{t=k}^{T} \left( y_t - \sum_{j=0}^{k-1} \boldsymbol{\beta}_{j+1}^T h(\mathbf{z}_{t-j}) \right)^2 + \lambda_1 \sum_{j=1}^{k} \left\|\boldsymbol{\beta}_j\right\|_1 + \lambda_2 \sum_{j=1}^{k} \left\|\boldsymbol{\beta}_j\right\|_2^2 .$$

The regularization is taken to be $\ell_2$ norm by default and the analysis with $\ell_1$ norm regularization and the effect of sparse modeling are discussed in *Appendix, Text*. Note that in our two-stage predictive model, the mapping to the feature vector is shared among all V4 neurons, whereas the top-layer linear model learned by regularized linear regression selects the specific CNN features for each neuron.

One concern of the CNN-based modeling approach is that conclusions about neuronal behavior based these models may be dependent on the details of the modeling procedure. To address this issue, we apply the stability principle [73] to interpret only the stable characterization from a large family of accurate predictive models. For each of the 71 V4 neurons, we fit models using features from layers 2, 3, 4 of the three pre-trained convolutional networks (AlexNet, GoogleNet, VGG), with either $\ell_1$ or $\ell_2$ regularization. This procedure results in 18 models in total (3 networks $\times$ 3 layers $\times$ 2 regression methods). In the next section, we propose *smooth DeepTune*, a stability-driven CNN-based model visualization framework to characterize V4 neurons' tuning preferences. This visualization is built upon finding the optimal stimulus for each neuron from each model and interpret the stable pattern across all models. Figure 3.1-D illustrates a summary of this framework. A stable interpretation of the DeepTune images from a pool of accurate models reveals the pattern selectivity of neurons. Subsequently, smooth DeepTune visualization is used to study the entire population of 71 V4 neurons.

Before presenting our visualization framework, we evaluate the prediction performance of our models. The correlation between the expected spike count predicted by the model and the actual average spike count on the holdout set is computed for all 18 models. As a baseline for comparison, we fit a V1-like model that extracts image features by applying a bank of linear Gabor wavelet filters to the input image at varying orientations, spatial frequencies, and phase, followed by half-wave rectification and a compressive nonlinearity.

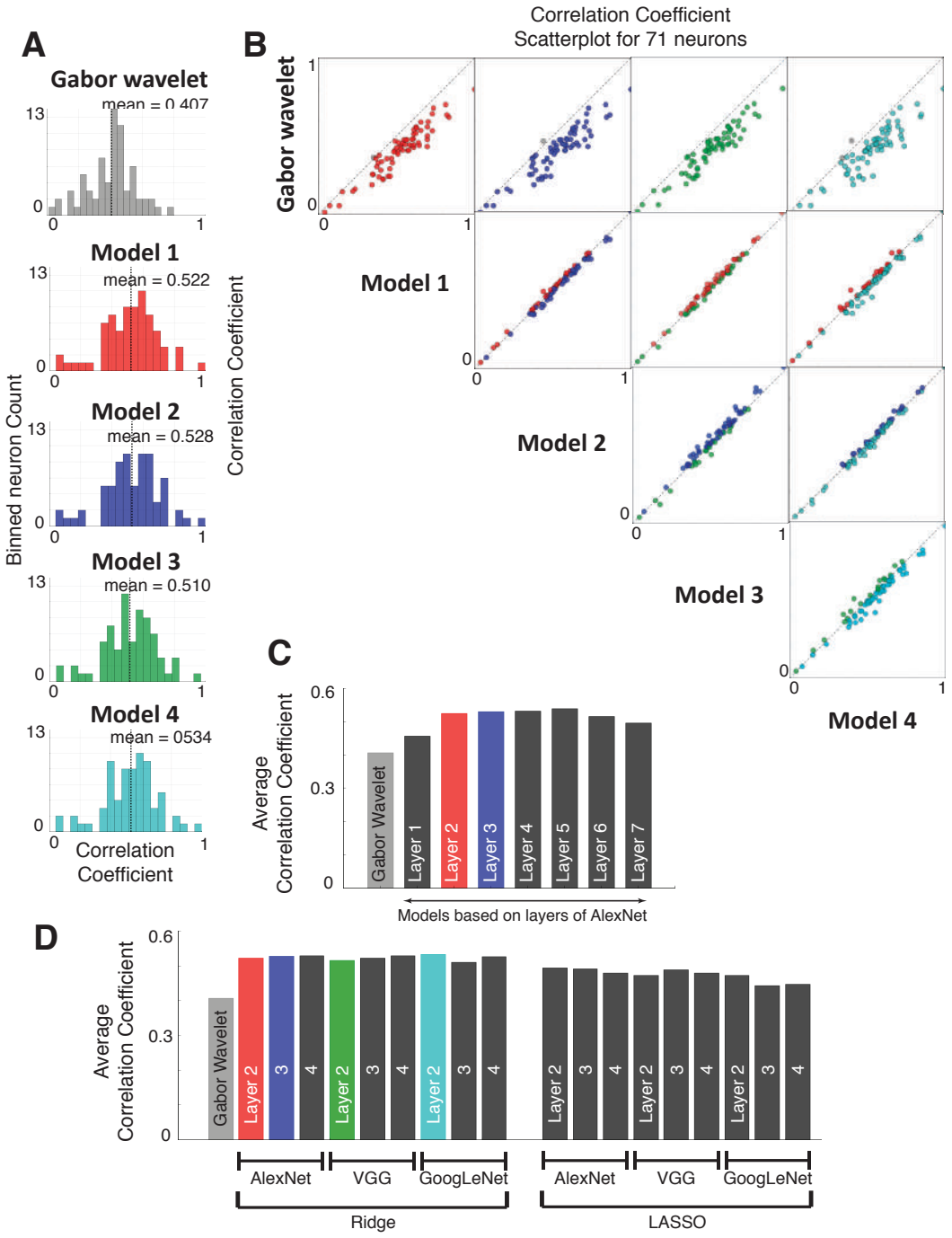


Figure 3.2: **CNN-based models have better prediction performance compared to Gabor wavelet model.** Noise-corrected correlation coefficient [60] is used as prediction performance measure in this figure. **A.** Histogram of correlation coefficients over population of V4 neurons for four models are shown in the first column. **B.** Scatter plots

comparing each pair of models. The four models shown here are: Model 1: AlexNet layer 2. Model 2: AlexNet layer 3. Model 3: VGG layer 2. Model 4: GoogleNet layer 2. Ridge regression is used in all models. **C.** The average prediction performance across 71 neurons for models from all 7 layers of AlexNet. The model based on Layer 1 of AlexNet has similar performance to that of Gabor wavelet model; while models from layers 2 to 5 have higher predictive performance. **D.** The average prediction performance across 71 neurons for all 18 models. All 18 models perform similar in prediction and much better than the Gabor wavelet model.

---

Our CNN-based models are highly predictive of the neuron spike-rate on the hold-out set. The AlexNet Layer 2-based model has an average correlation coefficient of 0.52 on the hold-out set, achieving the state-of-the-art accuracy for V4 neurons and comparable to that of V1 neurons. This is the noise-corrected correaltion computed based on the method described in [60]. The average raw correlation coefficient for this model is 0.44. All of the 18 models have average correlation coefficients higher than 0.51. Thehistogram of correlation coefficients over population of V4 neurons for four models are shown in Figure 3.2-A. These four models correspond to models based on (1) AlexNet Layer 2, (2) AlexNet Layer 3, (3) VGG Layer 2, and (4) GoogleNet Layer 2. The first two models are chosen in order to demonstrate the effect of the choice of layers and the last models are chosen to show the similarity difference between different neural networks. The scatter plots comparing each pair of models are illustrated in Figure 3.2-B. While all CNN-based models perform similarly, they are more accurate than the Gabor wavelet model for nearly all of the 71 V4 neurons. In Figure 3.2-C, we compare the prediction performance for models from all seven layers of AlexNet for averaged across 71 neurons. The model based on Layer 1 of AlexNet has similar performance to that of Gabor wavelet model; while models from layers 2 to 5 have higher predictive performance (0.52 for layer 2 to 0.54 for layer 5). Figure 3.2-D illustrates a complete comparison of the average correaltion coefficients between all 18 models. All of the CNN-based models perform similar in prediction and much better than the Gabor wavelet model. The LASSO-based models are in average slightly less accurate compated to Ridge-based models. However, the model size (in our case, the number of CNN features used in regression) is much fewer for LASSO-based models comapred to Ridge-based models. The median model size for LASSO is $\sim 750$, while it is $\sim 375,000$ for the Ridge regression.

The proportion of explainable variance captured by the model is an important metric for model prediction performance in the literature [55, 60]. It attempts to control for differences in noise levels between experimental setups, individual neurons, and brain regions. We estimate the explainable variance through the noise-corrected correlation co-

efficient [60] using the repeat presentations of images in the hold-out set (See *Appendix,* for more information). Averaged over the 71 V4 neurons, the AlexNet-based model captures 30% of the explainable variance. This is close to the variance explained by state-of-the-art of models of area V2 [66] However, there is still a slight gap between the performance of these models for area V4 and that of the state-of-the-art Gabor-wavelet models for area V1, which capture on the order of 40% of the explainable variance [13, 66].

## 3.3 Visualization of V4 neurons' pattern selectivity using stable smooth DeepTune images

Fully characterizing V4 neurons' tuning and pattern selectivity has been a difficult task, due to its highly nonlinear response properties [56]. It is expected that the high prediction performance of our CNN-based model should provide better characterizations of V4 neurons' tuning and pattern selectivity. However, unlike Gabor wavelet based or Fourier transform based models, the use of complex nonlinear CNN features renders the task of interpreting our models extremely difficult.

In this section, inspired by previous work in the attempt of visualizing and interpreting CNNs [74, 42], we introduce *smooth DeepTune images* as an representation of V4 neurons' tuning and pattern selectivity for an individual neuron. For each individual neuron, its smooth DeepTune images give rise to a collection of images that share some common visual patterns illustrating its pattern selectivity.

For each individual neuron, one DeepTune image is obtained by optimizing the input image stimulus, starting with a random image (e.g. white noise), such that our model output is maximized. This optimization is accomplished by applying regularized gradient ascent method from a random initial image until convergence. The regularization terms encourage the natural image found to be smooth and naturalistic. Formally, given our predictive model at a particular time lag $f : \mathbb{R}^{s\times s} \to \mathbb{R}$, we seek an input image stimulus $\mathbf{z} \in \mathbb{R}^{s\times s}$ that minimizes the following objective function:

$$-f(\mathbf{z}) + \lambda_p \mathcal{R}_p(\mathbf{z}) + \lambda_{\text{TV}} \mathcal{R}_{\text{TV}}(\mathbf{z}).$$

The regularization consists of two terms $\mathcal{R}_p$ and $\mathcal{R}_{\text{TV}}$. These two terms are designed to capture prior information about natural images by constraining the optimization search restricted to the smooth and naturalistic images [42]. The first regularizer $\mathcal{R}_p$ is defined as $\mathcal{R}_p(\mathbf{z}) = \|\tilde{\mathbf{z}}\|_p^p$, where $\tilde{\mathbf{z}}$ is obtained by mean-subtracting and flattening the image $\mathbf{z}$ into a vector. It encourages the intensity of pixels to stay bounded. By choosing a large $p$ ($p = 6$ in our analysis), this regualarizer prevents the solution from taking extreme pixel values.

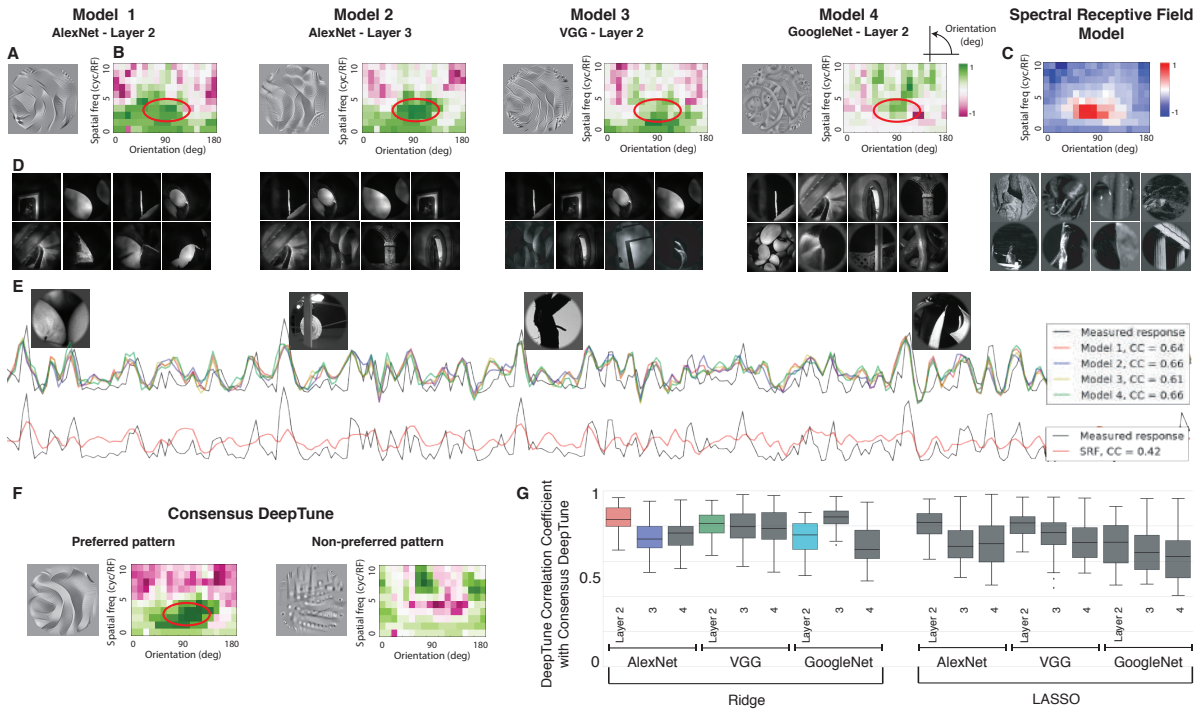


Figure 3.3: **For single neuron, smooth DeepTune images are stable across models. A.** Smooth DeepTune images based on four main models are shown for neuron 1. Note that these images share a visually stable curvature pattern with edges forming an approximately 90 degree angle (Also visible in the rest of our 14 models - see *Appendix, Figs. S8 and S9*). **B.** Power spectral density (PSD) of the smooth DeepTune images in polar coordinates. All four DeepTune images share a strong and stable frequency component in the range of 45 to 135 degrees with spatial frequency of 2 to 5 cycle per receptive field (the green color). **C.** Visualization of spectral receptive field (SRF) [12] model for neuron 1. The SRF visualization emphasizes in red the frequency components of the stimulus image selected by its regression model and therefore we use a different color-map (red-blue) than that for the DeepTune PSD image (green-pink). The pattern selectivity according to SRF is consistent with the stable parts of smooth DeepTune (highlighted in red circle). **D.** Images from training set with the highest responses for neuron 1. Similar curvature patterns to the smooth DeepTune visualization are visible in these images. **E.** The measured and predicted spike rates in the test set from the four main models as well as the SRF model for neuron 1. Images from the test set with the highest responses are visualized on top of the corresponding spike rate. Similar curvature patterns are visible in these images. Correlation coefficient between the measured and predicted spike rates is shown in the right panel. All of the four models outperform the SRF model. **F.** The consensus smooth DeepTune image for neuron A. Both preferred and non-preferred

patterns and the PSDs are shown. The preferred pattern exhibits the curvature contour that is stable among four models in panel A. The non-preferred pattern consists of lines orthogonal to the preferred curvature contour. The PSD confirms these observations. **G.** The box-plots of correlation coefficient between consensus DeepTune and all of the 18 DeepTune images from each model is shown across 71 neurons. DeepTunes from AlexNet layer 2 and GoogleNet layer 3 have the highest similarity to the consensus DeepTune.

---

The second regularizer $\mathcal{R}_{\text{TV}}$ controls the total variation norm of an image, encouraging the image to have piece-wise smooth parts and removing excessive and possibly spurious detail. This regularization is inspired by previous work in image denoising [57]. The finite-difference approximation of the 2-dimensional total variation norm on an image $\mathbf{z} \in \mathbb{R}^{s \times s}$ is defined as:

$$\mathcal{R}_{\text{TV}}(\mathbf{z}) = \sum_{(i,j) \in \{1,\dots,s-1\}^2} \left[ \left(\mathbf{z}_{i,j+1} - \mathbf{z}_{ij}\right)^2 + \left(\mathbf{z}_{i+1,j} - \mathbf{z}_{ij}\right)^2 \right]^{\frac{\alpha}{2}},$$

with $\alpha = 1$ in our analysis. Note that the smooth DeepTune optimization can also be carried out without any regularization terms (See *Appendix,*). The regularization effectively constrain the reconstruction space to the space of natural images.

Figure 3.3-A shows the smooth DeepTune images from four of our 18 models built for neuron 1. We verify that these smooth DeepTune images share a visually stable curvature pattern with edges forming an approximately 90 degree angle. While the rest 14 models exhibit slightly different smooth DeepTune images, the main curvature pattern stays relatively stable (see *Appendix, Fig. 3.D.3*). To quantitatively characterize the stable pattern, we compare the power spectral density (PSD) of these smooth DeepTune images in Figure 3.3-B. All four DeepTune images share a strong and stable frequency component in the range of 45 to 135 degrees with spatial frequency of 2 to 5 cycle per receptive field (the green color). Note that the high frequency components in Model 4 DeepTune image and the low frequency components in Models 1 to 3 are not stable across all models, therefore should not be interpreted. This stable characterization is consistent with the traditional computational models for V4 neurons. In Figure 3.3-C, we visualize the selectivity of frequency components for neuron 1 based on the spectral receptive field (SRF) [12] model. The SRF visualization shows the frequency components of the stimulus image selected by the regression model and therefore the color-map (red-blue) is chosen to be different from the DeepTune Fourier transform (green-pink). Similar to the interpretation from the stable smooth DeepTune, SRF confirms that neuron 1 exhibits a strong preference to the frequency component in the range of 45 to 135 degrees with spatial frequency of 2 to 5 cycle per receptive field. Similar curvature patterns are

visible in the images from training and test set with the highest responses for neuron 1 (Figure 3.3-D and E). Figure 3.3-E also illustrates the measured and predicted spike rates in test set from the four models as well as predicted spike rates by the SRF model. Our four models have similar and accurate prediction (correlation coefficient between 0.61 to 0.64), while the SRF model has difficulty in locating the peak spike rates (correlation coefficient of 0.42).

In addition to visual investigation of the stable patterns across 18 smooth DeepTune images generated from 18 models, we also introduce consensus smooth DeepTune image to represent the stable part in one image. The consensus smooth DeepTune image is obtained via similar optimization to original DeepTune with additional aggregating of gradient information from all 18 models. The aggregated gradient keeps the stable parts of gradients and averages out the unstable parts. (More details in *Appendix,*). The consensus smooth DeepTune image for neuron A is shown in Figure 3.3-F. Both preferred and non-preferred patterns as well as PSD of each image are illustrated in this figure. The preferred pattern exhibits the curvature contour also visible across all four models in Figure 3.3. The PSD confirms this observation by exhibiting strong frequency components in the range of 45 to 135 degrees with spatial frequency of 2 to 5 cycle per receptive field. On the other hand, the non-preferred pattern consists of lines orthogonal to the curvature contour. Some blobs are also visible in the non-preferred consensus DeepTune which suggests neuron A is suppressed by blob-like texture patterns. The corresponding PSD has strong high frequency components which is consistent with these observations.

To quantify the similarity of consensus DeepTune with each of the 18 model's DeepTune, we compute the correlation between images. Figure 3.3-G visualizes the box-plots of correlation coefficients across 71 neurons for each model. The median correlation for all of the models are considerably high (the highest median correlation is 0.83 which is achieved by AlexNet layer 2 and GoogleNet layer 3, both with $\ell_2$ regularization). Models with $\ell_1$ regularization tend to have lower similarity to the consensus DeepTune which is due to the sparsity of patterns in them. In the subsequent sections of this chapter and due to limited space, we present by default the consensus smooth DeepTune image as a stable representation of V4 neuron. Note that it is interesting to compare the 18 DeepTune images (available in the *Appendix, Figs. S9*) to verify that the consensus smooth DeepTune images keep the stable and interpretable part.

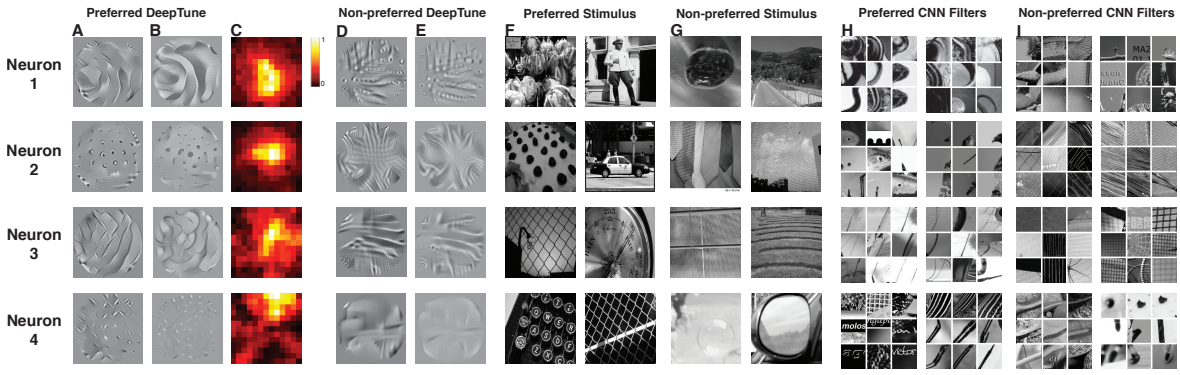


Figure 3.4: **A comparison of DeepTune images with average regression weight maps, selected CNN features for four V4 neurons and preferred stimulus (neurons 1, 2, 3, 4). A.** Preferred Smooth DeepTune images from AlexNet Layer 2 Model. **B.** Preferred consensus smooth DeepTune images based on all 18 models. Neuron 1 is tuned to the curvature shapes with edges forming an approximately 90 degree angle. Neuron 2 is selective for blob-like patterns and textures. DeepTune image for neuron 3 shows selectivity to curvature patterns with a strong diagonal line preference. Neuron 4 is tuned to corner-like shapes with edges forming 90 degree angles. The rest of the 17 models show consistent patterns as shown in these DeepTunes (See *Appendix, Fig. 3.D.3*). **C.** Average regression weight map based on AlexNet Layer 2 model. For each neuron, the average regression weight map also exhibits stable patterns across models and it highlights the receptive field of a neuron. **D.** Non-preferred Smooth DeepTune images from AlexNet Layer 2 Model. **E.** Non-preferred Smooth DeepTune images from AlexNet Layer 2 Model. **F.** Top two preferred image stimulus from ImageNet ILSVRC test dataset based on AlexNet Layer 2 Model activation. To obtain the preferred image stimulus, we feed $100,000$ images from ImageNet test dataset to the AlexNet Layer 2 Model and select the two images stimulus with highest model response. **F.** Top two non-preferred image stimulus from ImageNet ILSVRC test dataset based on AlexNet Layer 2 Model activation. **H.** Top two preferred CNN filters based on the filter importance. To visualize a convolutional filter from a CNN, we show the 9 top image patches from the ImageNet training set that has the highest filter responses (See *Appendix, Text* for more details). These 9 top image patches are representative of what this convolutional filter is computing [74, 72]. The Top two selected CNN filters validate the findings based on smooth DeepTune images. For example, Neuron 1 is tuned for curved-contour patterns according to smooth DeepTune images and its top CNN filters are those which activate on curvatures of similar shape. Neuron 2 is selective for blob patterns and the top CNN filters activates respectively on blob pattern and on a quarter of a blob pattern. **I.** Top two non-preferred CNN filters based on the filter importance.

## 3.4 Interpreting V4 models via selected CNN features

The stable smooth DeepTune images approach in the previous subsection treated our CNN-based model as a black box and used optimization and stability based analysis to directly characterize individual V4 neuron's tuning. In this section, we show that opening up the black box and interpreting the regression coefficients as well as CNN features can provide further information about V4 neuron's spatial receptive fields and pattern selectivity. These findings allow us to connect with previous study on the receptive fields of V4 neurons.

As an example, we examine the regression coefficients from the second-stage of our AlexNet Layer 2 model (See *Appendix, Fig. 3.D.5 and 3.D.6* for the visualization of other models). Coefficients with large magnitude indicate sensitivity of the neuron to particular image features. The AlexNet Layer 2 features are of size $256 \times 13 \times 13$. They consist of 256 different variety of convolutional filters that are spatially located on a grid of size $13 \times 13$. The corresponding regression weights at one time lag is of the same size. Formally, the regression weights at the first time lag $\hat{\boldsymbol{\beta}}_1$ is of size $256 \times 13 \times 13$. We examine the regression weights by asking the following two questions: which location of the image has the regression weights with large magnitude?; which type of convolutional filters contributes the most to the prediction performance?

To answer the first question, we define *average regression weight map* as the $\ell^2$ pooling of regression coefficient values for all features (across the different convolutional filters and the lag delays) at each location of the $13 \times 13$ grid. Formally, if we denote $\hat{\boldsymbol{\beta}}_{mijk}$ as the unflattened regression weights $\hat{\boldsymbol{\beta}}_k$, then the average regression weight map $\Phi \in \mathbb{R}^{13\times 13}$ is defined as follows,

$$\Phi_{ij} = \sum_{m=1}^{256} \sum_{k=1}^{k} \hat{\boldsymbol{\beta}}^2_{mijk}.$$

Figure 3.4-A and B show AlexNet Layer 2 Model's smooth DeepTune images and the consensus DeepTune images for four neurons. From AlexNet Layer 2 Model's smooth DeepTune images, neuron 1 is tuned to the curvature shapes with edges forming an approximately 90 degree angle. Neuron 2 is selective to blob-like patterns and textures. DeepTune image for neuron 3 shows selectivity to curvature patterns with a strong diagonal line preference and Neuron 4 is tuned to corner-like shapes with edges forming

90 degree angles. The rest of the 17 models show consistent patterns as shown in these DeepTunes (See *Appendix, Fig. 3.D.3*). This is also reflected by observing the similarity between the consensus DeepTune images and AlexNet Layer 2 Model's smooth DeepTune images. Figure 3.4-C shows the corresponding average regression weight maps for four neurons. For each neuron, the average regression weight map also exhibits stable patterns across models (See *Appendix, Fig. 3.D.6* for comparison between all the 18 models). The spatial receptive fields predicted by the average regression weight maps are consistent with our experimental conditions: all images are presented at CRF of the macaques. The spatial receptive fields are thus all close to the center of the image and they are roughly the size of one third of the whole image. But at the same time these receptive fields in V4 also come in diverse shapes. In Figure 3.4-B, the receptive field for neuron 1 and 2 has a circle-like shape, while neurons 3 and 4 form straight and curvature lines. The diversity in size and shape of the receptive fields of V4 neurons also validates previous studies on V4 neurons' diverse behavior in spatial invariance [45].

To address the second question on the contribution of convolutional filters, we calculate the filter importance of each of the 256 convolutional filters for each neuron by performing $\ell^2$ pooling of the regression coefficients for a particular convolutional filter across spatial locations. Formally, for each neuron, the filter importance $I_m$ of $m$-th convolutional filter in second layer of AlexNet is defined as follows,

$$I_m = \sum_{i=1}^{13} \sum_{j=1}^{13} \sum_{k=1}^{k} \hat{\boldsymbol{\beta}}^2_{mijk}.$$

To visualize a convolutional filter from a CNN, we use the CNN visualization technique introduced by [74] and show the 9 top image patches from the ImageNet training set that has the highest filter responses (See *Appendix, Text* for more details). These 9 top image patches are representative of what this convolutional filter is computing [74, 72]. Taking Neuron 1 as an example, Figure 3.4-C shows the top two filters among the 256 types of Layer 2 AlexNet features ranked by the feature importance $I_m$. Each selected feature is represented by 9 image patches. By visualizing these 9 image patches, we conclude, according to our model, that Neuron 1 is selective to curavture-type local patterns.

We also find that V4 neuron's curvature selection [45] could arise either due to systematic variation in fine-scale orientation tuning across spatial locations (This is case of Neuron 3), or due to local tuning heterogeneity (This is the case of Neuron 4). The latter category of neurons are interesting that they respond to curvatures at a finer scale, while respond to particular texture at a coarse scale. This is also consistent with earlier proposals, in which neuronal responses in V4 to combinations of line elements are weighted averages of the responses evoked by individual elements (Ghose and Maunsell, 2008; Lee

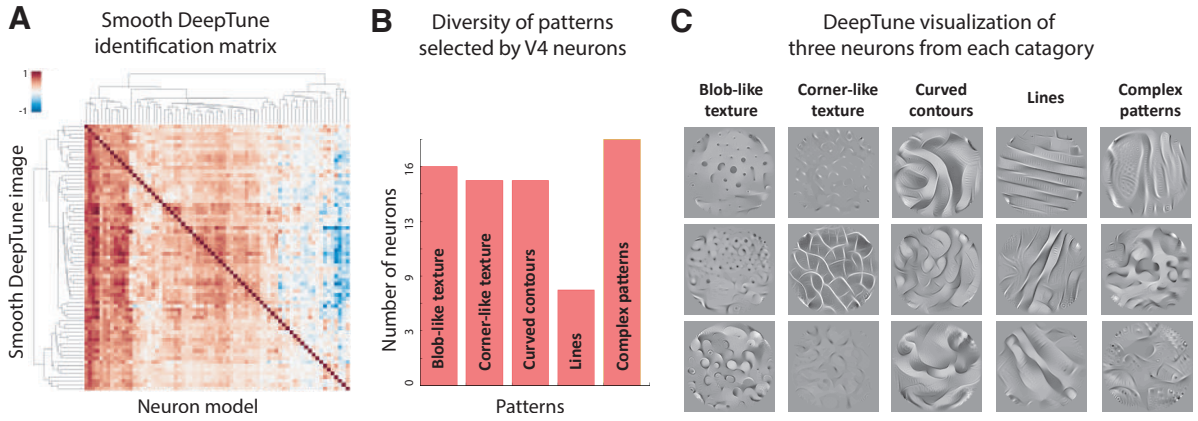


Figure 3.5: **Diversity of tuning among V4 neurons. A.** Responses of models for 71 neurons to all of the 71 consensus smooth DeepTune images are shown in the identification matrix. All models are based on AlexNet layer 2 with $l_2$ regularization in regression. A clear diagonal line is formed which demonstrates that for each neuron, the consensus DeepTune image has the highest response to its own model compared to that of other neurons' models. **B.** Neurons are manually categorized into five categories based on their smooth DeepTune images. More than 40% of the neurons are selective to texture patterns, half of which prefer blob-like textures and the other half prefer corner-like textures. About 30% of the neurons exhibit contour patterns, both curvature and straight lines. Neurons selective to curvatures are twice as the ones selective to straight lines. The rest of the neurons have selectivity to visually complex patterns. **C.** Examples of consensus DeepTune images for three neurons from each of the five categories.

and Maunsell, 2010). Note that this type of results would be difficult to obtain via models based on Fourier analysis such as spectral receptive field (SRF) [69, 12]. Because the 2D Fourier transform is space-invariant, it is difficult to distinguish for example Neuron 3 and Neuron 4 via SRF analysis.

## 3.5 Diversity of tuning in V4 to both contour and texture patterns

A number of studies have demonstrated that V4 neurons are selective for both visual shape properties (e.g. contour or curvature features) and surface properties (e.g. color or texture). However, the main functionality of V4 neurons as a population have been debated. Early synthetic shape stimuli based studies have shown that V4 neurons are tuned for orientation and spatial frequency of edges and linear sinusoidal gratings [16],

non-Cartesian gratings [19, 20] as well as curvature of contours [51, 50]. On the other hand, studies based on texture argue that shape properties might be seen as individual elements of a textured surface [4] and V4 plays a major role in processing textural information [43, 47].

In agreement with these studies, we show via smooth DeepTune images that the two types of neurons (more selective to shape properties or to texture properties) co-exist among the population of 71 V4 neurons in our experiment. To demonstrate this diversity, we first compute smooth DeepTune images for each neuron and then construct a response identification matrix. For each smooth DeepTune image image, we compute the responses from each of the 71 neuron models to it and plot them together in the identification matrix in Figure 3.5-A. The smooth DeepTune image for each neuron has the highest response to model of that neuron compared to other neurons in the population (diagonal line visible in Figure 3.5-A). No pair of the columns looks exactly identical shows that the 71 neurons' response properties are diverse.

We manually clustered these 71 neurons into five categories by looking at the smooth DeepTune images. Figure 3.5-B shows the histogram of these five categories and Figure 3.5-C shows smooth DeepTune images for three example neurons in each category. Both texture-tuned and contour-tuned neurons are present in area V4. In fact, from 71 neurons investigated in this study, about 40% of them are selective to textures and 30% percent of them prefer contours. A finer categorization shows that among the ones selective to textures, half of the neurons are tuned to blob-like patterns and the other half prefer corner-like patterns. Contour-selective neurons show preference to either curvatures or straight lines. Neurons selective to curvatures are twice as the ones selective to straight lines. We have also included in the last category the neurons tuned for complex patterns that are hard to describe in language and do not fall into previous categories. This observation agrees with previous studies [31, 56] that suggest V4's tuning to more complex shape properties.

## 3.6 V4 neurons' tuning to a wide range of separation angles

It is suggested [56] that curvature tuning in V4 helps provide an efficient way to encode shape. However, it is not clear how different types of curvature tuning are distributed. [51, 50] used stimuli constructed by joining two oriented line segments in a sharp corner or curve to study the shape selectivity in area V4 and showed the presence of curvature tuning at various orientations and separation angles.

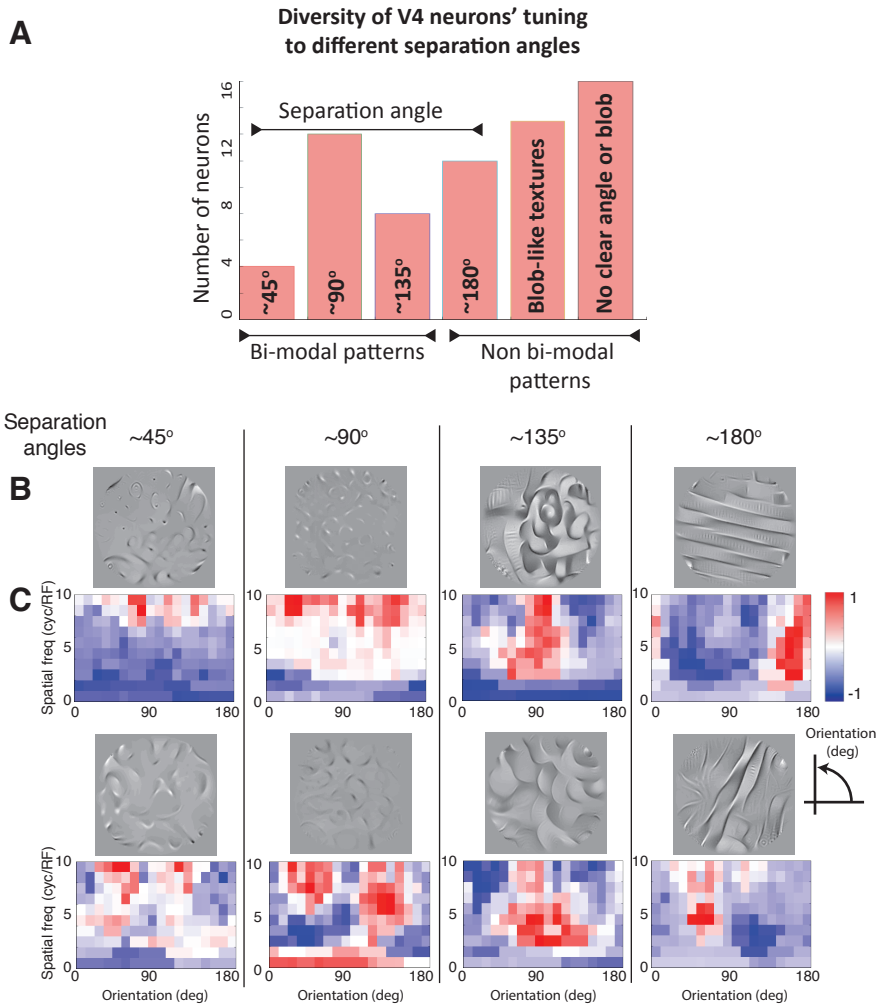


Figure 3.6: **Categorization of V4 neurons based on their separation angles. A.** Neurons are manually categorized into six groups. The first four groups contains neurons tuned to patterns with separation angles of 45°, 90°, 135°, and 180°. These patterns are either contours or textures. About 20% out of 71 neurons are tuned to patterns with separation angles of 90°. Another 20% of the neurons are selective to blob-like textures which does not correspond to any particular angle. The rest of neurons are not selective to any clear angle or blob-like patterns. **B.** The consensus smooth DeepTune image for two example neurons in each of the first four categories and **C.** the corresponding spectral receptive field (SRF) (David et al [12]) visualization. The orientation tuning obtained via SRFs and consensus DeepTune images are consistent. while SRF predicts a neuron has tuning for a particular angle through Fourier analysis, the consensus DeepTune images offer concrete visualization of these tunings. For example, for the bottom left neuron, both our method and SRF show an orientation tunings of about 70% and 120%.

The SRF based study in [12] also confirms this finding: the bimodal SRF is a consequence of V4's selectivity for sharp corners. Furthermore, Carson et al. [9] has shown via a sparse coding that not all curvatures are equally represented: there is a stronger representation of acute curvatures across the neural population.

In this section, we demonstrate that our CNN based model can account for selectivity for the curvature features and the smooth DeepTune Images provide concrete characterization of the separation angles. First we identify, visually via the consensus smooth DeepTune images, that V4 neurons have strong separation angles and corner shapes. Second, we manually classify these V4 neurons into four categories (45°, 90°, 135°, 180°) based on their separation angles [50, 12]. Figure 3.6-A shows a histogram of the V4 neurons in these four categories. We find that there is a strong presence of neurons having shape tuning at separation angles of about 90° (close to 20% of the 71 neurons). Another 20% of the neurons are selective to blob-like textures which does not correspond to any particular angle. The rest of the neurons are not selective to any clear angle or blob-like patterns.

To further demonstrate how consistent our smooth DeepTune images based classification of V4 neurons, we compare these findings with SRF analysis. In Figure 3.6-B and C, for each neuron, we display together the consensus smooth DeepTune image and the SRF plot based on the analysis of David et al. [12]. The horizontal axes of the SRF show the orientation tuning of each neuron, with excitatory component in red. We verify that the orientation tuning range and spacing shown in SRF are consistent with our smooth DeepTune images. For example, for the bottom left neuron, both our method and SRF show an orientation tunings of about 70° and 120°. The wide variability of preferred curved-contour patterns, matches the previous neurophysiological observations in V4. As in the SRF analysis [12] but on a different dataset, we find that V4 neurons have large orientation and spatial frequency bandwidth and often bimodal orientation tuning. Our smooth DeepTune images offer a concrete visualization of the bimodal orientation tuning properties of many neurons. For example, for the two example neurons shown in the 90° column, consensus smooth DeepTune image illustrates the bimodal orientation tuning concretely by showing its preferred pattern which looks like a corner.

## 3.7 Suppressive tuning in the cortical area V4

It is well known that V4 neurons have surround suppressive mechanisms [16, 59, 32] just like many other visual cortical areas [27, 3]. Recent study [66] has also reported evidences for the presence of strong suppressive tuning for half of the neurons in area

V2, which is not caused merely by surround suppression. Willmore et al. [66] also discovered that this type of suppression is not present in area V1. It is natural to question whether such suppressive tuning is also present in the area V4. To study the suppressive tuning in V4 area, we fit models of V4 neurons based on the Berkeley Wavelet Transform (BWT) [66], adopt the same excitation index defined in the same paper to rank the neurons in V4 and finally visually confirm the suppressive neurons via smooth DeepTune images. The BWT-based model provides a nonlinear spatio-temporal receptive field (STRF) for each neuron (More information in *Appendix, Fig. 3.F.1*). The excitation index (EI) is defined as:

$$EI = \frac{\Sigma h^+ - \Sigma h^-}{\Sigma h^+ + \Sigma h^-}$$

where $h^+$ and $h^-$ are the excitatory and suppressive weights assigned to the wavelets in each STRF, respectively.

The BWT-based model has an average prediction correlation coefficient of 0.33 for the 71 V4 neurons in the hold-out test set. While this performance is 0.11 lower than the best CNN-based model, it is high enough to study the suppression tuning in V4 (the same model achieves an average correlation coefficient of 0.30 for V2 neurons [66]). Figure 3.7-A shows the histogram of excitation index for 71 V4 neurons. 41% of the neurons show suppressive tuning. The median of excitation index for V4 neurons is 0.10. While, the portion of neurons with suppressive tuning is 9% lower compared to V2 neurons, it is 29% higher compared to neurons in cortical area V1 [66].

The smooth DeepTune visualization enables us to validate the suppressive tuning in the V4 area. Figure 3.7-B illustrates the excitatory and inhibitory smooth DeepTune images for three neurons identified as the most suppressive neurons by the BWT model and the corresponding excitation index. Here, the excitatory smooth DeepTune images are obtained via maximizing the model response, while the suppressive ones are obtained via minimizing the model response. The neuron excitation index and response of the model to each smooth DeepTune image are shown in the same panel. The smooth DeepTune images give a concrete visualization of the suppressive tuning. The neurons' excitatory DeepTune images are largely blurred, while their suppressive DeepTunes shows more clear patterns. For neuron 43, while the excitatory DeepTune has unstable and relatively weak patterns, the suppressive DeepTune exhibit a stable tuning to 90 degree corner shapes in the right hand side of visual field. Neuron 43 model response to the excitatory DeepTune image ($r = 0.54$) is relatively low compared to other neurons (Excitatory neurons have responses as high as $r = 31$). Both neurons 27 and 26 have strong suppressive tuning to complex shapes with mid range frequency, while their excitatory DeepTunes have weaker patterns compared to the excitatory neurons.

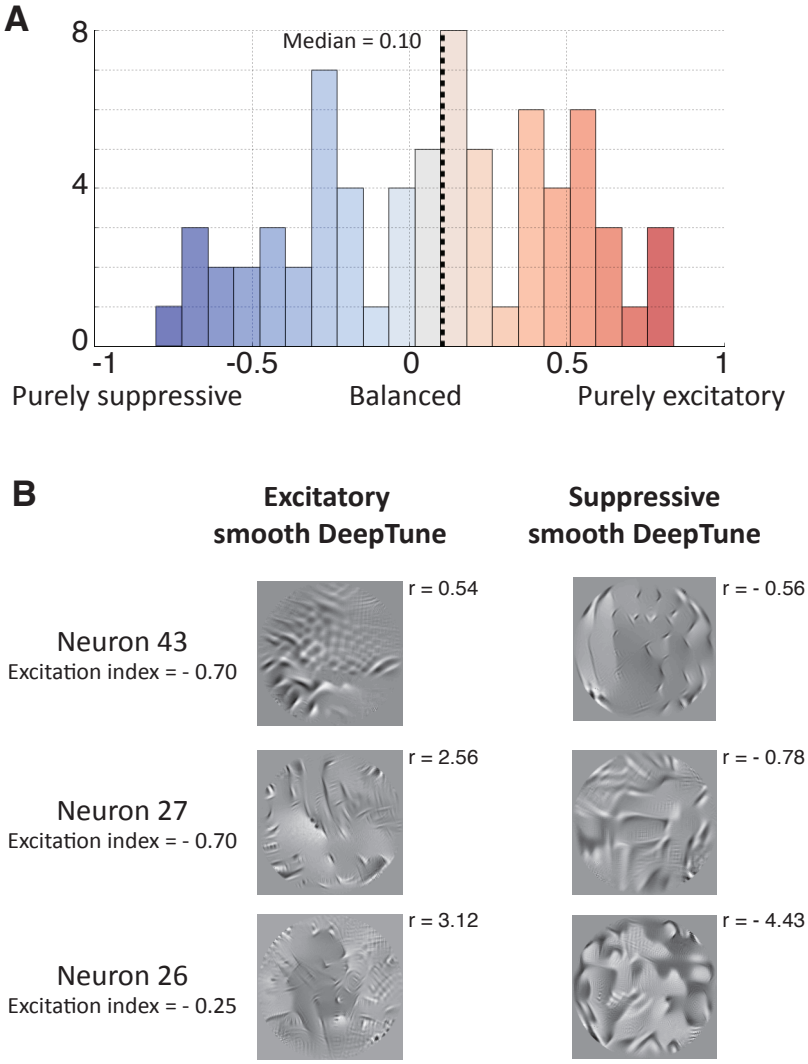


Figure 3.7: **Neurons in cortical area V4 exhibit suppressive tuning. A.** Histogram of excitation index for 71 V4 neurons. 41% of the neurons show strong suppressive tuning. The median of excitation index for V4 neurons is 0.10. **B.** The excitatory and inhibitory smooth DeepTune images for three neurons identified as suppressive by the STRF model and excitation index. The neuron excitation index and response of the model to each smooth DeepTune image is illustrated in the same panel. The neurons with suppressive tuning tends to have a clearer suppressive DeepTune visualization than those without.

## 3.8 Discussion

In summary, we proposed state-of-the-art CNN-based models for V4 neurons in predicting neural spike rates in a hold-out validation set to natural image stimuli. Our smooth DeepTune visualizations provide stable and concrete characterization of V4 neurons. The population analysis shows that V4 area is tuned a huge variety of curvature contour patterns as well as textures in different orientations. We also brought the analysis of suppressive neurons proposed in [66] to a new level by providing concrete visualization of how suppression function happens in suppressive neurons.

#### DeepTune visualization as a general basis for single-neuron tuning visualization

The idea of computationally optimizing input stimulus to find one neuron's tuning pattern dates back to Carlson et al. [9], where the evolutionary sampling method was used to optimize for the stimulus that causes the highest number of spikes. This approach has largely expanded the search space of tuning patterns when compared to previous work based on hand-crafted stimulus patterns. While the evolutionary sampling method is constrained on concatenated Bezier splines, our DeepTune optimization has an even larger search space and allows for more complicated tuning patterns. Provided that we have a differentiable predictive model of single neuron, the DeepTune visualization serves as a general approach for visualizing optimal tuning, which can also be applied to other visual areas in future work.

#### DeepTune visualization could serve as input stimulus for future neurophysiology experiments

We are able to provide more concrete visualization of V4 neuron's receptive field than previous studies. On one hand, studies based on hand-crafted stimuli [19, 20, 51] might lack the expressive power to represent all V4 neurons' receptive field. On the other hand, predictive modeling approaches such as SRF [12] only provide summary statistics about the receptive fields. It would be difficult to reconstruct the tuning patterns of V4 neurons solely from its spatial frequency tuning and orientation tuning summary based on SRF. The DeepTune images we provide for each V4 neuron are close enough to the space of input image stimuli. It remains in future experiments to use these DeepTune images as input image stimuli to verify the predicted V4's tuning properties in this work.

**Possibility to obtain simple description for all V4 neurons' tuning properties?**

In this study, we have shown that V4 neurons are tuned to a huge variety of curvature contour and texture patterns via DeepTune visualization. Looking through the DeepTune visualization for all the V4 neurons, we observe that not all tuning patterns shown in DeepTune images are simple: even the stable part of some V4 neurons' smooth DeepTune images are difficult to describe in natural language (e.g. See Figure 3.7). This suggests that area V4's tuning is much more complicated than that of V1 neurons as well as what previous studies on V4 suggest via hand-crafted grating stimuli. Many of them might not be stimulated via simple stimuli inputs. In future experiments, it would be interesting to investigate this hypothesis by manually design stimuli as complicated as what we have shown in this study.

**Natural Stimuli**

In this study, we have investigated V4 neurons' behavior using natural stimuli. While these stimuli present serious challenges both for neurophysiology experiments and statistical analysis and modeling, they are essential for understanding how the brain represents and processes visual input outside the laboratory. Natural stimuli probe a large portion of the space of a neuron, while hand-crafted stimuli design is inherently human-biased. Producing a model that can accurately describe the responses of the neuron throughout this high-dimensional space is challenging.

Additionally, it has been observed in [66] that natural stimuli also have important advantages for understanding the interaction between excitation and suppression. Simple stimuli such as might not be able to to stimulate neurons to produce both excitation and suppression.

# Appendix

## 3.A Data collection

In this paper, we have used extracellular recordings from well isolated neurons in parafoveal areas V4 (71 neurons) of two awake, behaving male rhesus macaques (Macaca mulatta). This dataset has been previously used to study the sparseness of neural codes in the area V4 [67]. For the details of the data collection procedure refer to [67].

All procedures for the neurophysiology data collection are approved by the Animal Care and Use Committee of the University of California, Berkeley. The procedures were conducted in strict accordance with good practice as defined by the Office of Laboratory Animal Care at UC Berkeley, the National Institutes of Health, the Society for Neuroscience, and the American Association for Laboratory Animal Science.

The training dataset of a neuron consists of $4,000 - 12,000$ natural images. Spike count was measured at 60 Hz, resulting in two measurements per image. For the held-out dataset, 300 images were shown for each neuron, distinct from the images shown for the training dataset. The sequence of test images was repeated; on average, each image in the test set was shown 9.3 times. The resulting spike counts were averaged to provide a lower-variance estimate of the expected spike count; repeats also allowed for estimation of the amount variance in the neuron explainable by the stimulus image (signal-to-noise).

## 3.B Modeling methodology and metrics

In this section, we discuss the methods we used to model neurons and the metrics to characterize our models' performance compared to measured neuron activity. As described in the main text, we use transfer learning framework to analyze sinlge V4 Neuron input-response data: We first extract convolutional neural networks (CNN) features and then use as predictors in a linear regression method to predict spike-rates as the response. The CNNs are pretrained on large scale image classification dataset ImageNet [58]. The linear model learned by regularized linear regression is trained on our data.


Figure 3.A.1: **Sample of Images from training and holdout datasets. A.** 50 images sampled from training dataset of 4000 images of one neuron. **B.** 25 images sampled from holdout dataset of 300 images of one neuron.

As a measure of the prediction performance of our model, the correlation between the expected spike count predicted by the model and the actual average spike count on the holdout set is computed.

*Explainable variance* captured by the model is another relevant metric for predction performance in the neuroscience litterature [55, 60]. This metric attempts to control for differences in noise levels between experimental setups, individual neurons, and brain regions. We estimate explainable variance using the repeat presentations of images in the test set.

## Convolutional Neural Networks (CNNs)

Deep convolutional neural networks are a successful tool to analyze big data problems and are therefore are being actively studied for a vast variety of applications especially in machine learning [34, 33, 61].

Convolutional networks are basically neural networks with several layers and a specialized connectivity structure. The purpose is to extract features of the scene in multiple layers. It has been shown that higher layers compute more global features than lower layers, so that the hierarchical structure provides a better overall quality of features [74]. The proposed architecture for several layers of network varies in different applications but it usually consists of three general types: convolutional layers, pooling layers and fully-connected layers.

*Convolutional layers* select a window of previous layers output and convolve it with a set of filters. Dependencies are local in this structure. The coefficients of these filters

are tunable weights of our network and their final value will be specified in the training procedure. As an example, considering images as the input of our network, each filter is a rectangular grid which will be convolved with specific patch of previous layer. A non-linear function will be used to specify the output of the neuron as in traditional neural networks. Equation 3.1 specifies this relationship between output of different layers for two-dimensional configuration.

$$y_{i,j}^{l} = f\left(\sum_{m=0}^{M}\sum_{n=0}^{M} w_{mn} y_{i+m,j+n}^{l-1}\right) \tag{3.1}$$

where $i$ and $j$s indicates possible spatial location at layer $l$, $y_{i,j}^{l}$ is the output of each neuron in layer $l$, $w_{mn}$ is the filter weights at location $(m,n)$ of layer $l-1$ and $f$ is the non-linear function.

*Pooling layers* could be utilized after each convolutional layer. It simply performs a spatial pooling over patches of previous layer. These patches could be overlapping or non-overlapping. The output of pooling layer for each patch is a single value which in most of the cases is maximum value of the patch. Pooling could be useful to reduce the feature dimension as well as increase the invariance of the features for small transformation. It also helps to increase the size of receptive field for each feature value.

After several convolutional and pooling layers aimed at grasping the low-level and high-level features, a few *fully-connected layers* are used as the final stages of the network. These layers are essential for specific application of the network such as classification or prediction.

Figure 3.B.1 shows the neural network architecture of the AlexNet model [33]. It consists of five convolutional layers, three pooling layers inbetween and two fully connected layers. Our analysis is carried out on all the seven layers shown in the figures. Layers L2, L3 and L4 are of the main focus. In particular, the output feature at L2 is of size $256 \times 13 \times 13$, where 256 indicates the number of types of filters applied at Layer L2, $13 \times 13$ indicates that the features are extracted on a spatially-equally-spaced $13 \times 13$ overlapping grid of the original image. Similarly the output features at layers L3 and L4 are of size $384 \times 13 \times 13$ and $384 \times 13 \times 13$.

We also used GoogLeNet [63] and VGG [62] in our analysis. The architectures and the mechanism of these models are beyond the scope. We refer the readers to the original paper for a detailed understanding. The CNN feature extraction pipeline is done using the Caffe [29] package and the model files provided within.

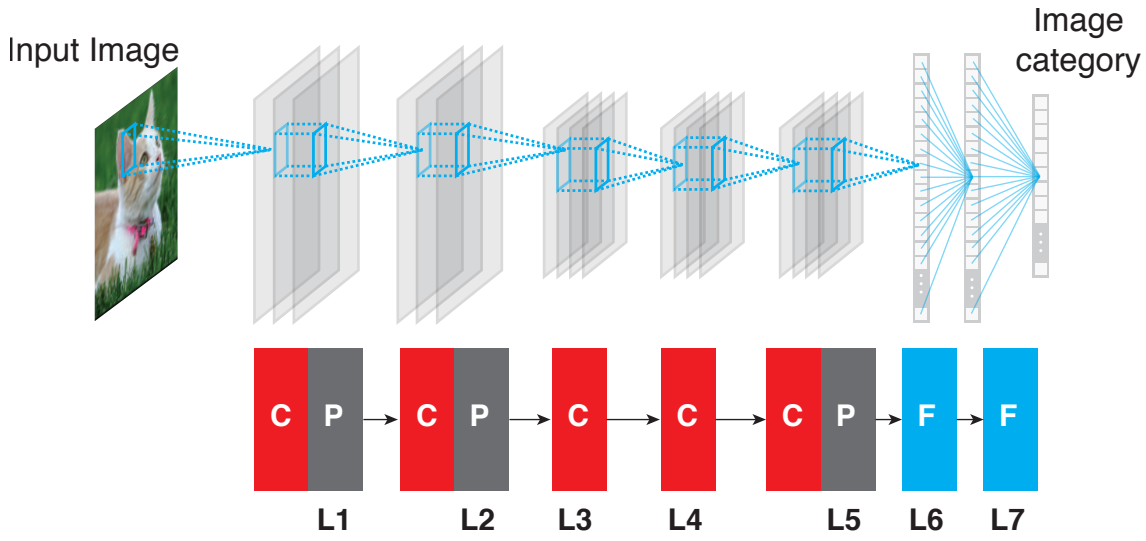


Figure 3.B.1: **Architecture of the AlexNet model [33].** Red box indicates convolutional layer, gray box indicates pooling layer and blue box indicates fully connected layer.

## Regularized linear regression methods

As described in the main text, our predictive model for a single neuron response takes the following form

$$F : \mathbb{R}^{s\times s\times k} \to \mathbb{R}$$
$$(\mathbf{z}_t, ..., \mathbf{z}_{t-k+1}) \mapsto \sum_{j=0}^{k-1} \boldsymbol{\beta}_{j+1}^\top h(\mathbf{z}_{t-j}),$$

where $(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k) \in \mathbb{R}^{d\times k}$ are the regression parameters to be determined and $h$ is the fixed CNN feature transform.

To perform the regression analysis, we solve the following regularized linear regression problem

$$\left(\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_k\right) = \underset{\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_k}{\arg\min} \frac{1}{2} \sum_{t=k}^{T} \left( y_t - \sum_{j=0}^{k-1} \boldsymbol{\beta}_{j+1}^\top h(\mathbf{z}_{t-j}) \right)^2 + \lambda_1 \sum_{j=1}^{k} \left\|\boldsymbol{\beta}_j\right\|_1 + \lambda_2 \sum_{j=1}^{k} \left\|\boldsymbol{\beta}_j\right\|_2^2.$$

Taking the AlexNet Layer 2 model as an example, the Layer 2 feature is of dimension $d = 256 \times 13 \times 13$. Taking into account the time lags, the weight matrix $\left(\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_k\right)$ is of dimension $256 \times 13 \times 13 \times 9$. This feature dimension is much larger than the sample

size $T = 8000$. Regularization methods are needed to both improve prediction accuracy and provide better interpretation.

Either ridge regression ($\ell_2$ regularization) or LASSO [64] ($\ell_1$ regularization) will be suitable for this high dimensional regression problem. While ridge regression is commonly used in the neuroscience litterature, LASSO could provide better guarantees for feature selection [75] in theory. We find that both regression methods produce consistent prediction performance and smooth DeepTune images in our analysis. A detailed comparison is discussed in Section 3.D.

## 3.C Visualization of CNN filters

In this subsection, we provide visualization of CNN filters. These visualizations show that CNN features encode much richer patterns that Gabor wavelets do. They support our finding that CNN based models perform better than simple Gabor wavelet based models in modelling V4 neurons. Because of the pooling operations, normalization operations, and non-linear activation functions in the CNNs, the CNN features are complex nonlinear functions of the raw input image. These CNN features are the outcome of learning from the large scale image dataset ImageNet, and are in general hard to explain via mathematical formula.

Inspired by the recent advances in CNN visualization [74, 72], we visualize the filters Layer 2, 3 and 4 of the AlexNet as follows: taking Layer 2 as an example, for each of the 256 types of filters, we exaustively search for nine image patches, from a dataset of one million image patches generated from ImageNet, that has the maximal output responses for the filter. The one million image patches are generated by randomly cropping images in ImageNet.

Figure 3.C.1 shows a subset of 256 types of filters in Layer 2 of AlexNet. We have manually clustered these filters in categories. We observe that other than encoding edge-shape patterns, Layer 2 of AlexNet also encodes a rich set of curvature patterns, contour-blob patterns as well as crossing patterns. These patterns could be very useful in building a predictive model for V4 neurons, because similar shape tuning properties of V4 neurons have been reported before [56].

Similarly, Figure 3.C.2 and Figure 3.C.3 shows a subset filters in Layer 3 and Layer 4 of AlexNet. We observe that these filters encode even richer shape patterns. Some concrete patterns such as “dog head” and “birds” appear in Layer 3 filters. It has been shown that higher layers compute more global complex features than lower layers [74]. Unfortunately, the higher layer features become more specific to the classification task used to train AlexNet. It is not clear the higher layer features are as transferable as the

lower layer features to other tasks [61].


Figure 3.C.1: **Visualization of subset of filters in Layer 2 of AlexNet.** To visualize each filter, we have fed one million image to the CNN and visualized top nine image patches that activate that has the maximal output responses for the filter [74]. We have manually clustered filters into categories.


Figure 3.C.2: **Visualization of subset of filters in Layer 3 of AlexNet.** To visualize each filter, we have fed one million image to the CNN and visualized top nine image patches that activate that has the maximal output responses for the filter [74]. We have manually clustered filters into categories.


Figure 3.C.3: **Visualization of subset of filters in Layer 4 of AlexNet.** To visualize each filter, we have fed one million image to the CNN and visualized top nine image patches that activate that has the maximal output responses for the filter [74]. We have manually clustered filters into categories.

## 3.D Stability Analysis

In this section, we discuss the stability of our analysis for smooth DeepTune images and model selected features.

### Stability of smooth DeepTune images

Our main analysis is based on smooth DeepTune Images. The CNN-based approach for interpretation is potentially biased because of the specific choice of architecture, parametrization and methods.

#### Convergence

To visualize the smooth DeepTune image optimization process, we use SuperHeat visualization package to plot the heatmap of the CNN feature activation map throughout the optimization process in Figure 3.D.1. There is a transition of the CNN feature activation map at about smooth DeepTune iteration 8. After this iteration, the CNN feature activation map stabilizes. The inactive columns correspond to the color-selective features in AlexNet. Our stimulus is gray-scale, therefore, it is expected to observe weak selection for these filters.

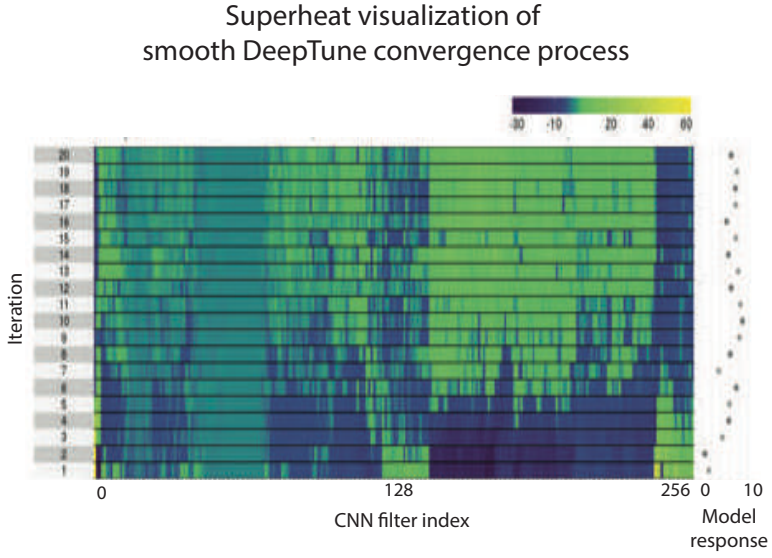


Figure 3.D.1: **The smooth DeepTune image optimization process.** We use SuperHeat visualization package to plot the heatmap of the CNN feature activation map throughout the optimization process.

### Stability across different initialization

A DeepTune image is the final result of an optimization process on an initial random image. To study the effect of random initialization on the final DeepTune image, we run the optimization process on 10 different random starting image for each neuron. Figure 3.D.2 shows 10 DeepTune images from these different initializations for five neurons. The patterns from 10 DeepTune images are visually similar. The average pair-wise correlation coefficient between 10 images is 0.95 for neuron 1. For other neurons, this value is not less than 0.91.

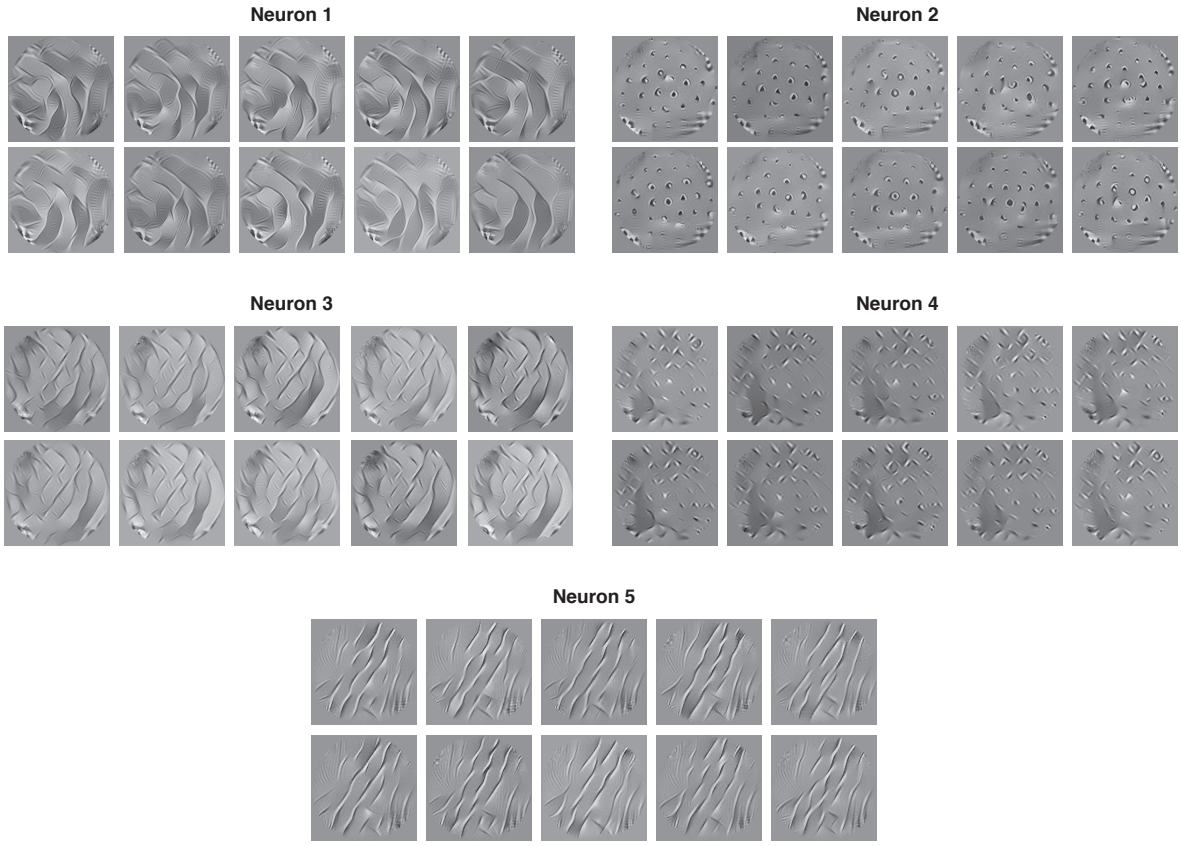


Figure 3.D.2: **Stability of smooth DeepTune images with 10 different random initializations for five neurons.**

### Stability across 18 models

The smooth DeepTune images from all of the 18 models studies in this paper has stable patterns for each neuron. To construct the 18 models, we have used 3 pre-trained convolutional neural networks (AlexNet, VGG, and GoogleNet). From each network, we use either two, three, or four layers to extract features from images in neuroscience experiments. These features predict the spike rates of each neurons using a regularized linear regression. We use both $l_2$ (ridge regression) and $l_1$ (LASSO) regularizations. This results in 18 models for each neuron (3 networks, 3 layers, and 2 regression model). Figure 3.D.3 shows smooth DeepTune images from each of these 18 models for two neurons. The stable pattern among these 18 DeepTune should be interpreted as the pattern that activates neuron.

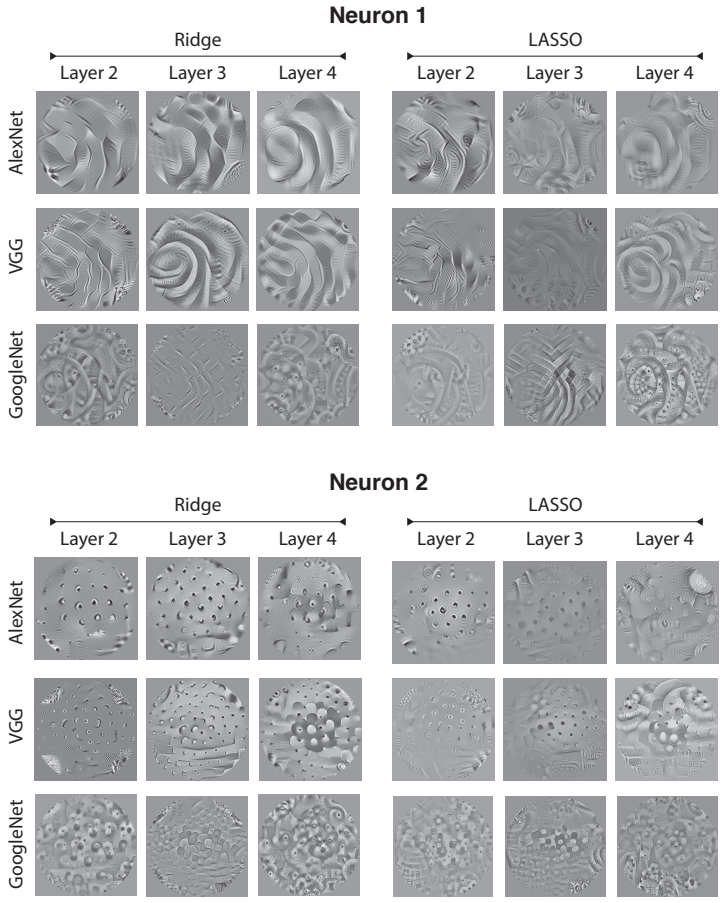


Figure 3.D.3: **Stability of the interpretable patterns in smooth DeepTune mages for neurons 1 and 2 across 18 models.**

### Identification matrix across models 1, 2, and 3

To demonstrate this diversity, we first compute smooth DeepTune images for each neuron and then construct a response identification matrix. For each smooth DeepTune image image, we compute the responses from each of the 71 neuron models to it and plot them together in the identification matrix in the top heatmap plot in Figure 3.D.4. The smooth DeepTune image for each neuron has the highest response to model of that neuron compared to other neurons in the population (diagonal line visible in Figure 3.D.4). No pairs of columns looks exactly identical which is an evidence that the 71 neurons' response properties are diverse. We also study the stability of this observation by feeding Deeptune images generated from VGG and GoogleNet models to AlexNet-based model.

Figure 3.D.4 the middle and bottom heatmap plots illustrates the responses of neuron models from AlexNet layer 2 to DeepTune images generated by VGG and GoogleNet layer 2 models. Ridge regression have been used in all of the models. The heatmaps in Figure 3.D.4 contain clear diagonal patterns, showing the smooth DeepTune images are stable across models. This observation, quantitatively confirms the visually observed stability of smooth DeepTune images.

## Stability of selected features and weight-maps

In this section, we investigate the stability of CNN features selected by each neuron across different models. First, we visualize the top selected features and show that these features have stable visualization across models. Then, we use the regression coefficients in models to identify the model-inspired receptive field for each neuron. This is achieved by visualizing heatmaps of average regression coefficients across all features corresponding to each location in image.

### Stability of top selected features across four main models for four neurons

Our model for each neuron consists of a CNN-based feature selection module and a linear regression model to predict the neuron spike rate from those features. Figure 3.D.5 shows that these features have stable visualization across models. For neurons 2, 3, 4, and 5, we visualize the filters representing top two selected features. Each box with 9 image patches visualizes a filter in the CNN. To visualize the filter, we feed a million random natural images (from AlexNet dataset) to the network and show the top 9 image patches that activate the filter. For each model the left box corresponds to the top filter and the right box corresponds to the second top filter selected by neuron. The patterns are stable across all four models.

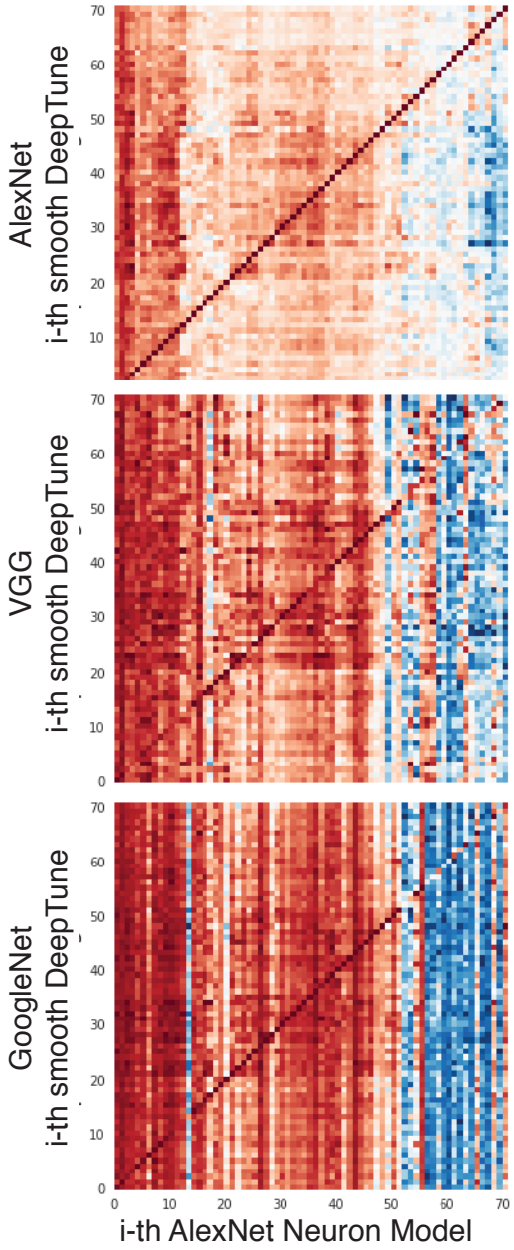


Figure 3.D.4: **Stability of model responses to smooth DeepTune images generated from AlexNet, Vgg, and GoogleNet based models.** Smooth DeepTune images from layer 2 of AlexNet, VGG and GoogleNet are generated for each neuron. These images are fed into our prediction model based layer 2 feature of AlexNet. All three heatmaps contain clear diagonal pattern, showing the smooth DeepTune images are stable across models. This observation, quantitatively confirms the visually observed stability of smooth DeepTune images.

For neuron 2, both top and second top filters for four models are selective to blob-like patterns. CNN filters selected by Neuron 3 like both curvatures and diagonal edges with 45 deg patterns. Neuron 3 prefers filters selective to corners and edges in both diagonals, however, no curvature filter is selected by this neuron. Neuron 5 is consistently selecting filters responsive to diagonal patterns in 45 deg. Similar observation holds for other V4 neurons on the population.

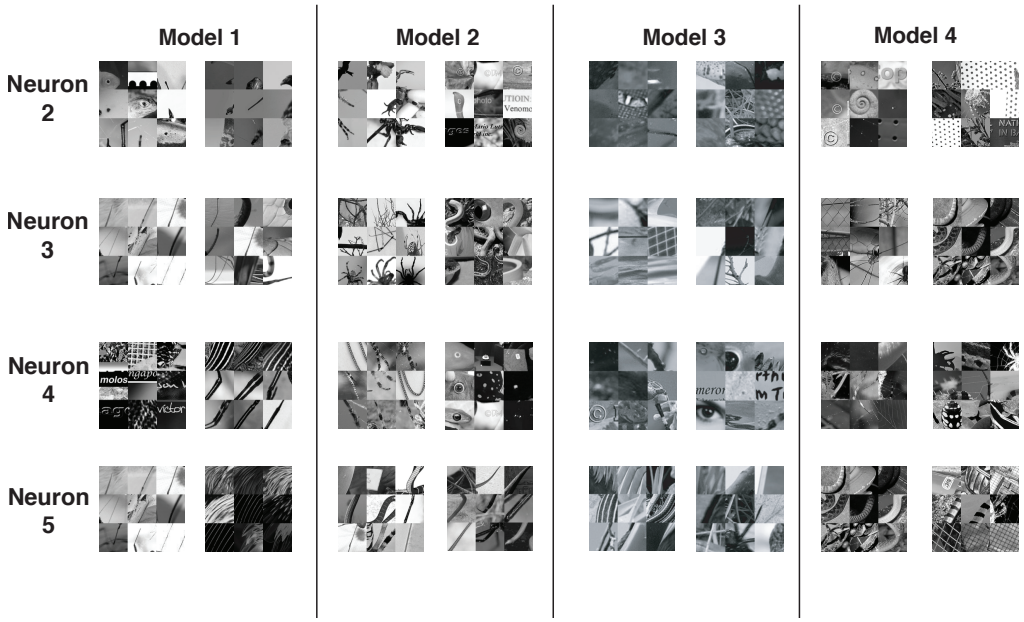


Figure 3.D.5: **Stability of top selected CNN features selected by each neuron across four main models.** Each box visualizes a filter representing the feature in the CNN. To visualize the filter, we feed a million random natural images (from AlexNet dataset) to the network and show the top 9 image patches that activate the filter.

### Stability of average weight-maps across four main models

In this section, we study the stability of model-inspired spatial receptive fields for each neuron. The CNN features extracted from images have spatial structure due to nature of convolution operation. That is, each feature corresponds to a location on the image. Consequently, the regression coefficients mapping these features to neuron spike rates have similar spatial structure. After fitting the predictive models for each neuron, we estimate a model-inspired receptive field for each neuron, by averaging the regression coefficients in each location across different filters. The heatmap of average regression coefficients for each location on the image represents the importance of that location for the neuron.

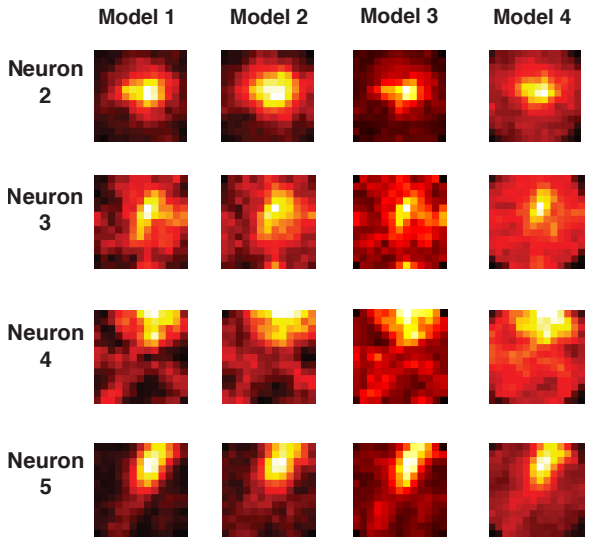


Figure 3.D.6: **Stability of average model weight-maps across four main models.** These weight-maps estimates the model-inspired receptive filed. After fitting the predictive models for each neuron, we estimate a model-inspired receptive field for each neuron, by averaging the regression coefficients in each location across different filters. Each row corresponds to one neuron. Each column corresponds to a model. The weight-maps are stable across models for each neuron.

Figure 3.D.6 shows these weight-maps for four neurons and four models. The weight-maps are stable across models. For neuron 2, the features in the center leaning to right side of the image are selected by the neuron. Neuron 3 and 5 are selective to features in a diagonal location. Neuron 4 prefers features in a cross-like location.

### Stability of LASSO vs. Ridge

Inspecting raw coefficients from models learned by Lasso is problematic, however, due to the instability of the Lasso selected features. Particularly in cases where regressors are highly correlated (as is the case with features extracted from a CNNs), the model selection performed by Lasso may be inconsistent [75]. To overcome this issue and focus on the truly salient features for a particular neuron, we performed a stability analysis using 10-fold cross validation: the model was refit on each of the 10 perturbed datasets, and then the sets of selected variables were intersected. Model coefficients on this set were then averaged and used as a basis for our analysis. This is similar to the method introduced in [5], except we use cross validation instead of bootstrap resampling.

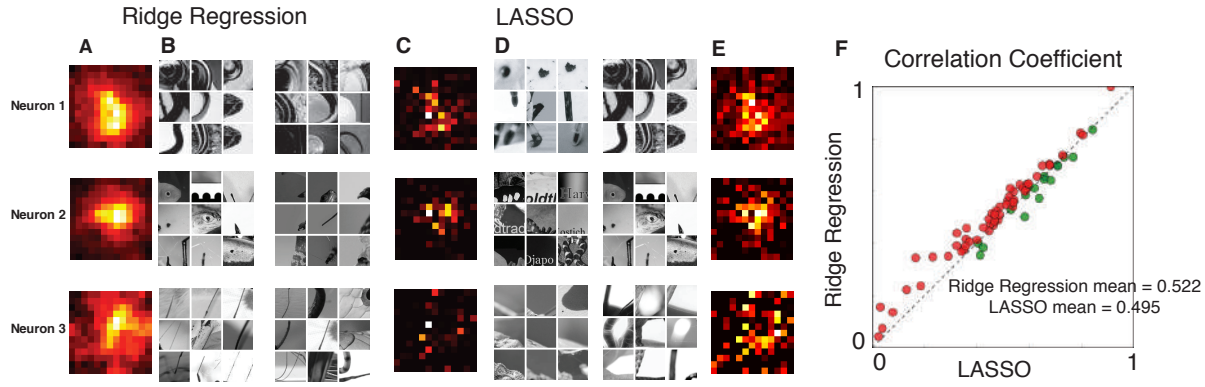


Figure 3.D.7: **Comparison of Lasso and Ridge feature selection.** Ridge and Lasso give similar prediction performance. Lasso in general selects a smaller number of features (751 features in average) included in the set of features that Ridge selected (total 377,000 features). The top CNN filters that Lasso selected are similar to that of the Ridge regression.

# 3.E DeepTune visualization of all V4 neurons in the population

In this section, the preferred and non-preferred DeepTune images are visualized for all of the 71 V4 neurons in the population.

## Preferred smooth DeepTune images for all 71 V4 neurons

Figure 3.E.1 shows the preferred DeepTune images for all of the 71 neurons under study in visual area V4. The model used here is AlexNet layer 2 with ridge regression. Refer to the main text for a discussion on the diversity of the patterns preferred by V4 neurons.

## Non-preferred smooth DeepTune images for all 71 V4 neurons

Figure 3.E.2 illustrates the non-preferred DeepTune images for all of the 71 neurons. The model used here is AlexNet layer 2 with ridge regression. Most of the neurons have weak patterns in their non-preferred DeepTune image. For some of the neurons, the pattern is stronger. In the main text of chapter 3, we present a detailed discussion on the interpretation of patterns in non-preferred DeepTune images.

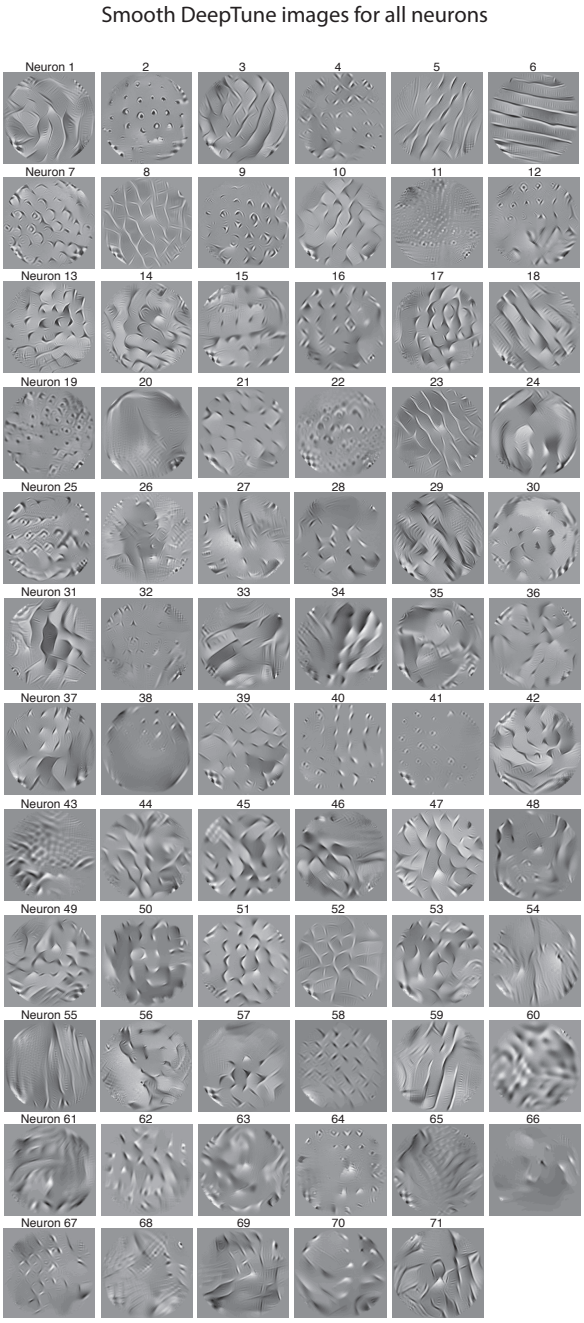


Figure 3.E.1: Preferred smooth DeepTune images generated by AlexNet-based model for all 71 V4 neurons.

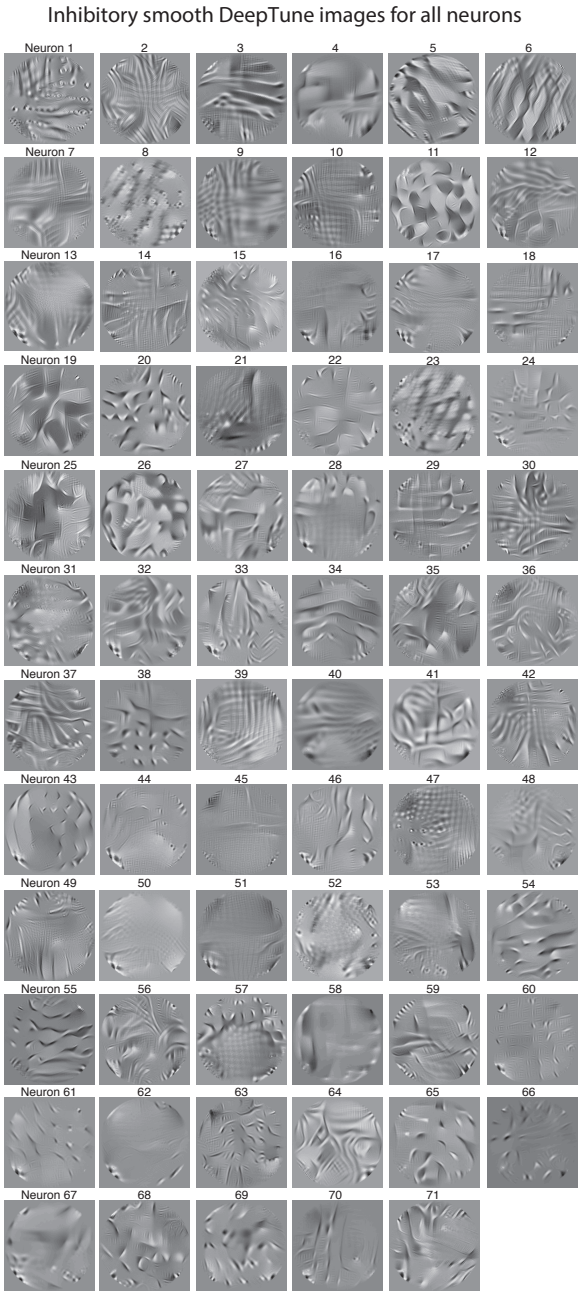


Figure 3.E.2: Non-preferred smooth DeepTune images generated by AlexNet-based model for all 71 V4 neurons.

## 3.F Spectral Receptive Field (SRF) model

It has been shown in [12] that many V4 neurons have more than one excitatory orientation tuning peak. Bimodal orientation tuning explains previous observations of selectivity for sharp corners [51]. Curvature or corner patterns will result in Bimodal orientation tuning in V4. We show via smooth DeepTune that a large part of V4 neurons share this property and the result is consistent with that obtained by the spectral receptive field (SRF) [12]. Details of the SRF model are discussed in the main text.

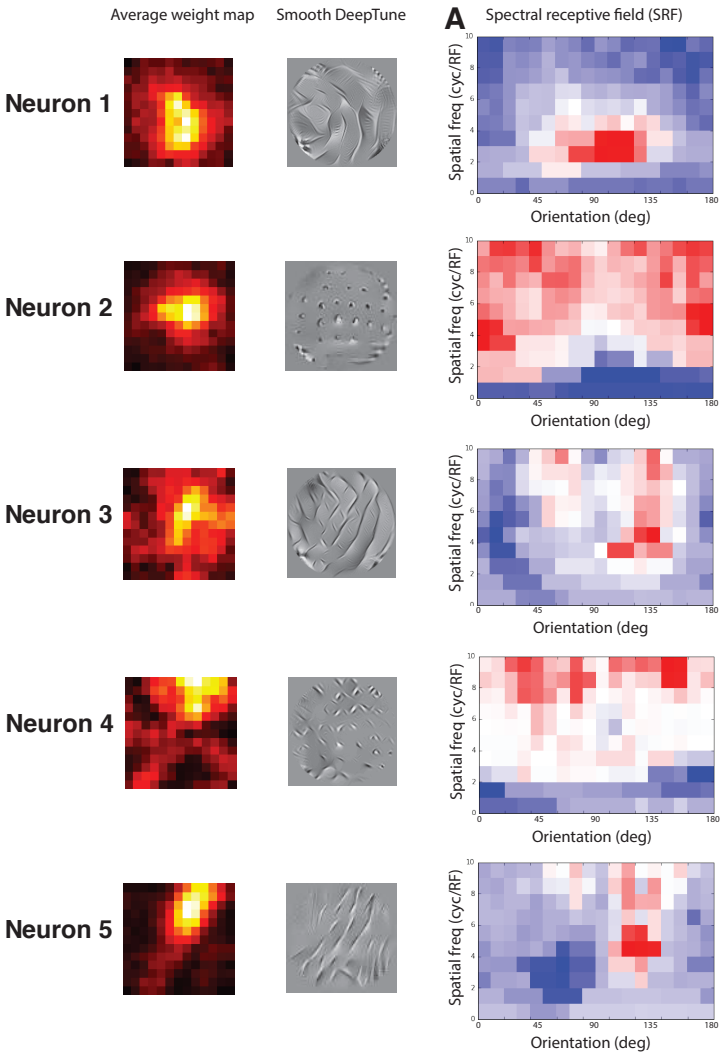


Figure 3.F.1: **Consistency of the average weight map and smooth DeepTune images with spectral receptive field (SRF) [12].** Details of the SRF model are discussed in the main text.

## 3.G Principle component analysis

Each V4 neuron model corresponds to a point in $p$-dimensional coefficient space; we can investigate the population of V4 neurons by examining their relative positions in this space. However, because $p$ is very large (in the case of models based on N2 of AlexNet, $p = 389,376$) direct analysis of the coefficient vectors is impossible due to the curse of dimensionality. First, we perform $\ell^2$ pooling of coefficient values across space and time delays to yield a single impact value for each of the filters in layer N2 of AlexNet. This gives a 256-dimensional representation, where each dimension corresponds to a single filter. Next, we perform principal components analysis (PCA) of the 71 points (each corresponding to a single V4 neuron) in this 256-dimensional space. PCA finds a set of linear transformations that capture a large proportion of the variance of the vectors. An examination of the coefficients of the loading vectors reveals that the first several principal components delineate several recognizable image features. The first principal component specifies whether neuron is selective to horizontal and vertical patterns. The second principal component delineates low-frequency patterns vs. dense blobs. The third principal component delineates diagonal vs non-diagonal smooth features.

Figure 3.G.1.A shows the plot of the 71 V4 neurons according to their values in the first two principal components. Each neuron is shown via its smooth DeepTune image. The color of DeepTune image borders is proportional to the third principle component with red being the highest PC value and blue the lowest. The neurons with highest values in PC 1 are selective to Figure 3.G.1.B illustrates the 71 V4 neurons according to other principal components. The coefficients of the loading vectors for first three principal components are shown in Figure 3.G.1.C. For the top coefficients, the corresponding filter is visualized by top 6 image patch that activate that filter. These image patches are found by feeding one million random image to the CNN and selecting the patches with highest filter response.

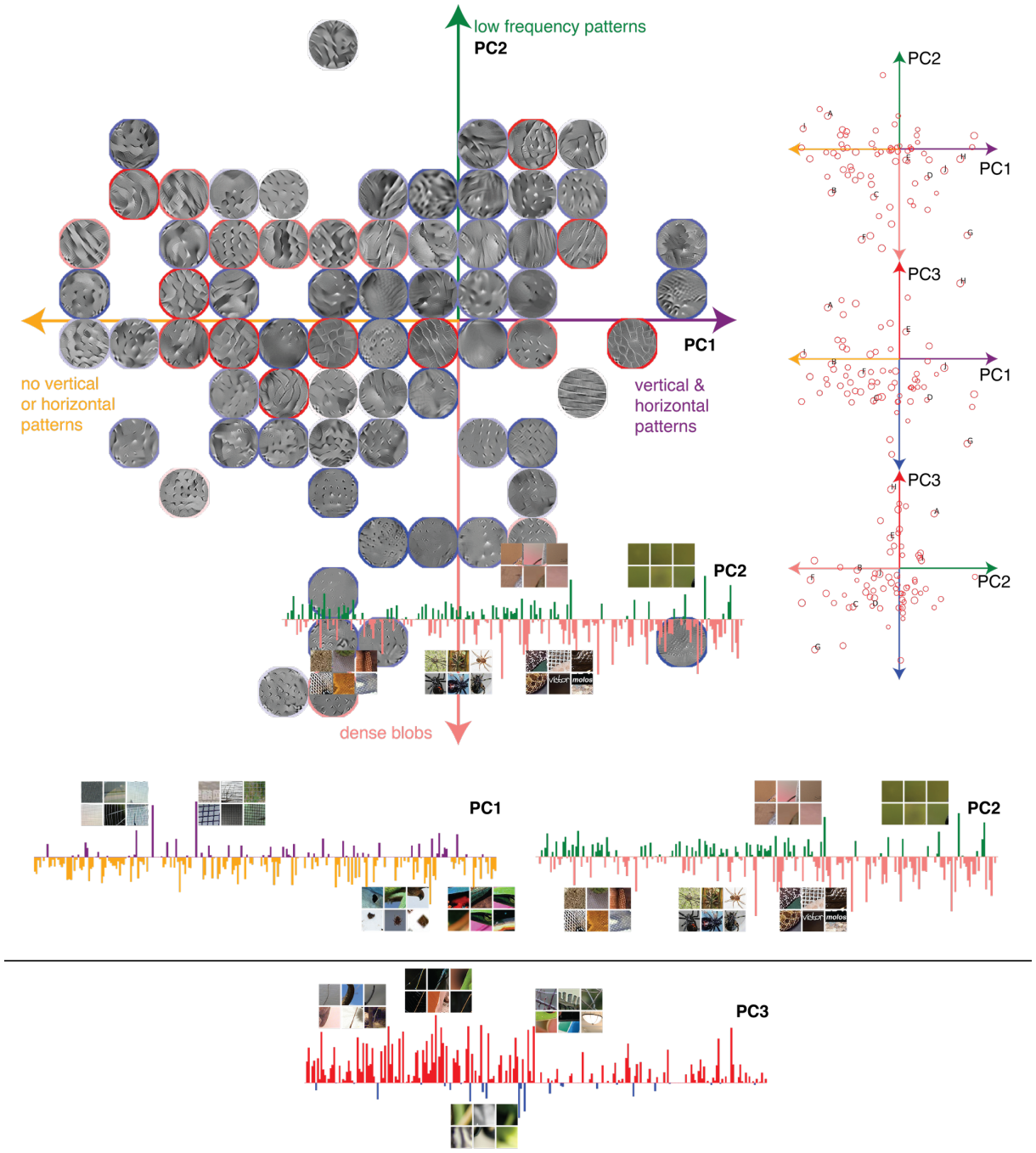


Figure 3.G.1: **Principle components analysis of V4 neuron's population. A.** 71 V4 neurons according to their values in the first two principal components. To compute the principle components, we perform $\ell^2$ pooling of coefficient values across space and time delays to yield a single impact value for each of the filters in layer N2 of AlexNet. This

gives a 256-dimensional representation, where each dimension corresponds to a single filter. Then, we perform principal components analysis (PCA) of the 71 points (each corresponding to a single V4 neuron) in this 256-dimensional space. Each neuron is shown via its smooth DeepTune image. The color of DeepTune image borders is proportional to the third principle component with red being the highest PC value and blue the lowest. **B.** 71 V4 neurons according to other pairs of principal components. **C.** Coefficients of the loading vectors for the top three principal components. For the coefficients with highest values, the corresponding filter is visualized by top 6 image patch that activate that filter. These image patches are found by feeding one million random image to the CNN and selecting the patches with highest filter response.

---

## 3.H Predicted responses of V4 models to hand-crafted stimuli

In this section, we investigate the response of our CNN-based neuron models to polar, hyperbolic, and Cartesian gratings. We manually create images in each category based on the equations give in [19]. The response of the V4 models based on second layer of AlexNet with Ridge regression are computed for each of these hand crafted images. Figures 3.H.1, 3.H.2, 3.H.3, and 3.H.4 show the responses of three neurons to these images. The smooth DeepTune image is also shown in each figure. The hand crafted images selected by the model are consistent with the patterns visible in the DeepTune image.

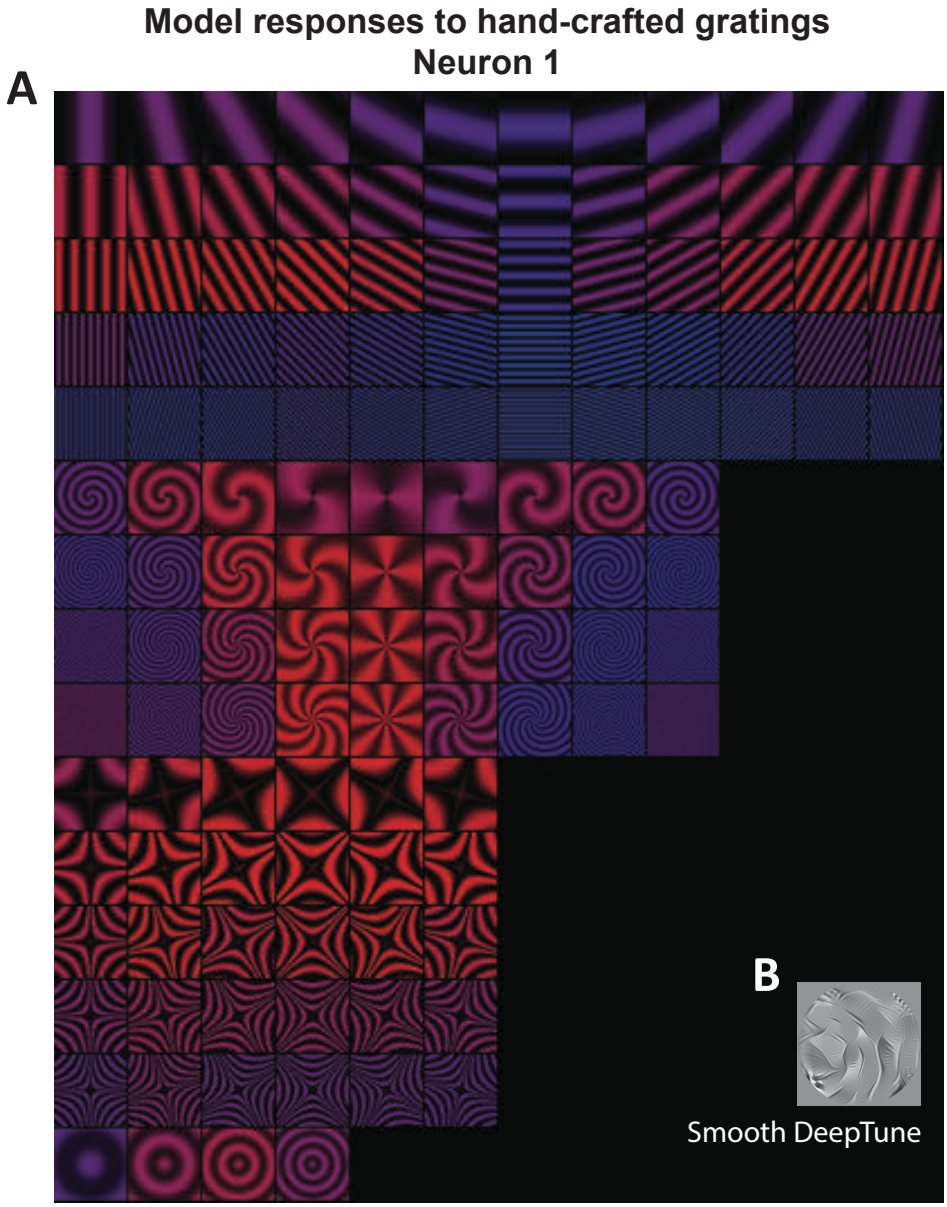


Figure 3.H.1: **Neuron 1 model responses to polar, hyperbolic, and Cartesian gratings are consistent with smooth DeepTune patterns. A.** Responses of neuron 1 model to polar, hyperbolic, and Cartesian gratings. The gratings in red and blue correspond to excitatory and inhibitory stimulus, respectively. **B.** Smooth DeepTune of neuron 1

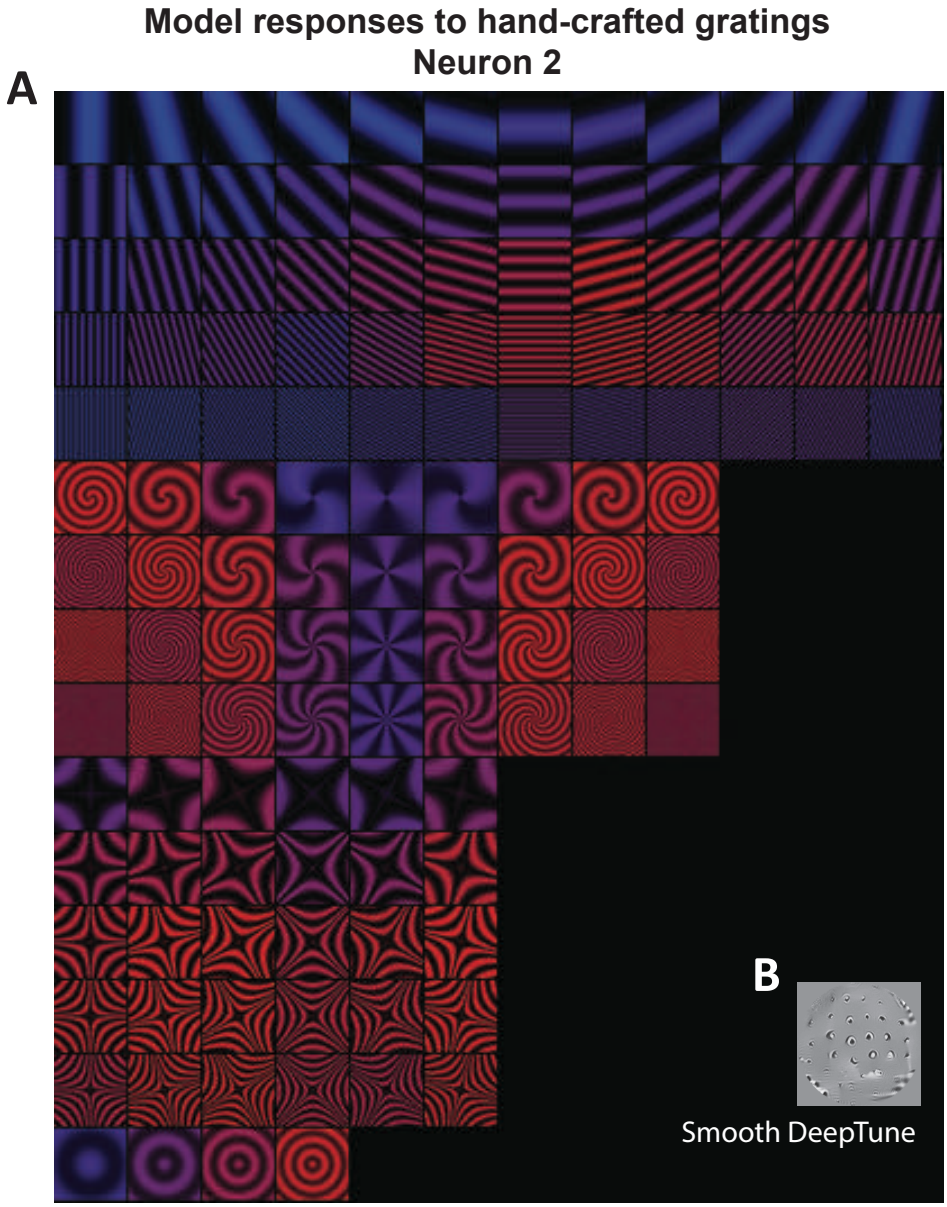


Figure 3.H.2: **Neuron 2 model responses to polar, hyperbolic, and Cartesian gratings are consistent with smooth DeepTune patterns. A.** Responses of neuron 2 model to polar, hyperbolic, and Cartesian gratings. The gratings in red and blue correspond to excitatory and inhibitory stimulus, respectively. **B.** Smooth DeepTune of neuron 2

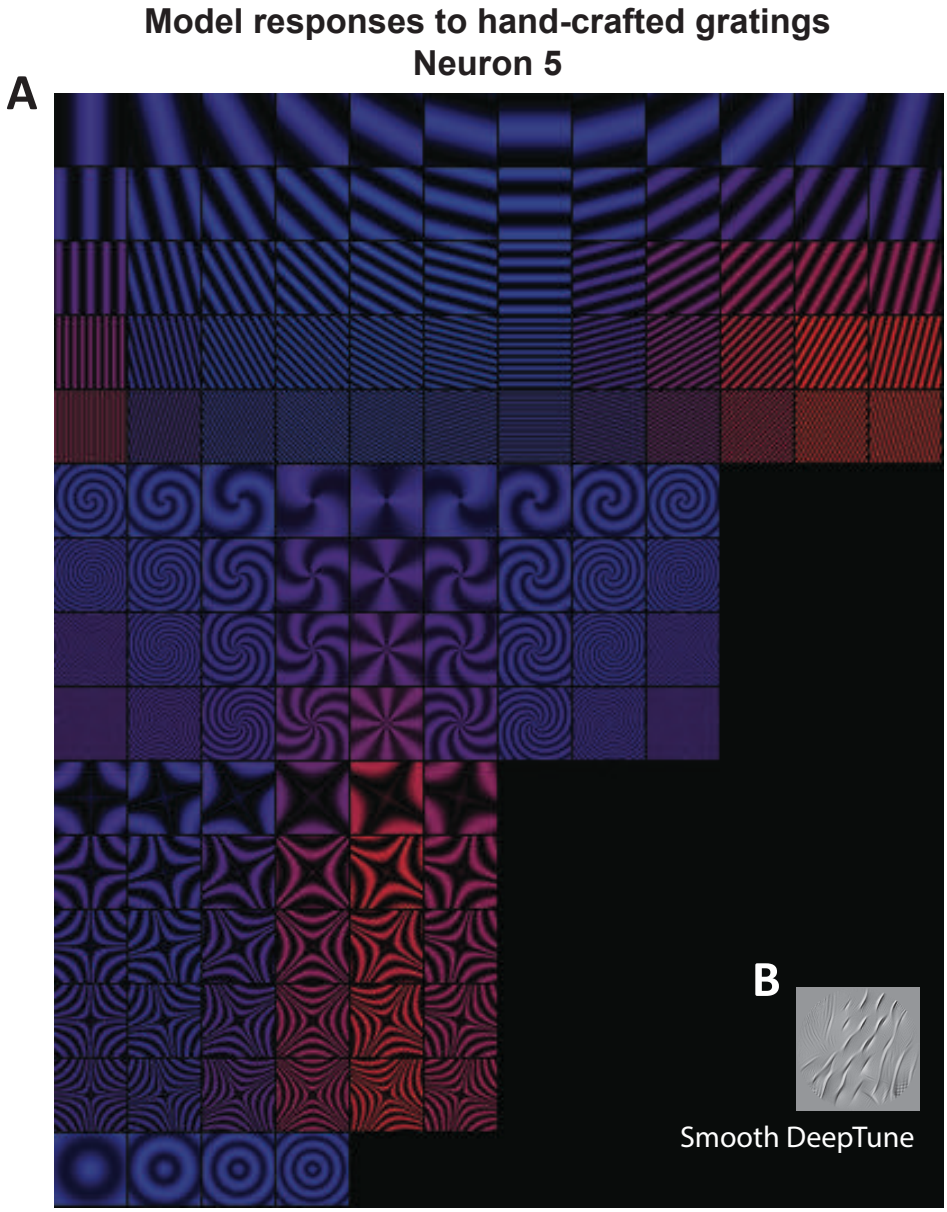


Figure 3.H.3: **Neuron 5 model responses to polar, hyperbolic, and Cartesian gratings are consistent with smooth DeepTune patterns. A.** Responses of neuron 5 model to polar, hyperbolic, and Cartesian gratings. The gratings in red and blue correspond to excitatory and inhibitory stimulus, respectively. **B.** Smooth DeepTune of neuron 5

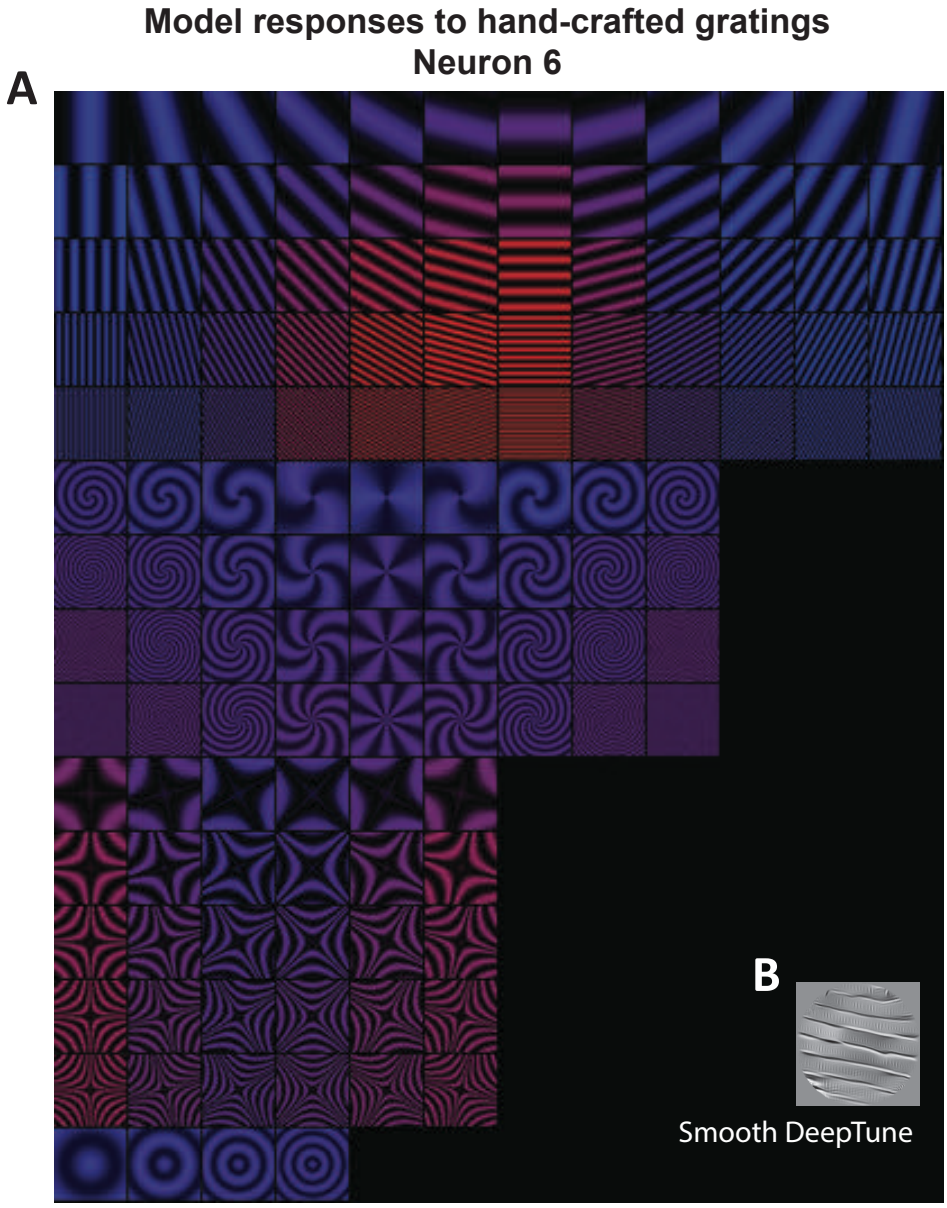


Figure 3.H.4: **Neuron 6 model responses to polar, hyperbolic, and Cartesian gratings are consistent with smooth DeepTune patterns. A.** Responses of neuron 6 model to polar, hyperbolic, and Cartesian gratings. The gratings in red and blue correspond to excitatory and inhibitory stimulus, respectively. **B.** Smooth DeepTune of neuron 6

# Chapter 4

# Compressed models of V4 neurons

## 4.1 Introduction

An important question in neuroscience is to describe the simplest, accurate model for a neuron. Producing simple representations has been challenging for CNN-based models, introduced in Chapter 3, but the number of filters used by a CNN provides a useful starting point in considering model simplicity. That is, a CNN with fewer filters is easier to interpret and visualize.

In Chapter 2, we provide a compression framework for convolutional neural networks based on pruning filters. In this chapter, we borrow similar ideas to build compressed models of neurons in visual area V4. Particularly, two importance indices is employed to perform compression: one is based on Classification Accuracy Reduction (CAR) and the other is based on Regression Accuracy Reduction (RAR). To prune CNNs using the CAR index, we employ our greedy compression algorithm introduced in Chapter 2. The CAR algorithm prunes filters in CNN based on their contribution to the classification accuracy. The pruned filters are transfered to predict spike rates of V4 neurons. In this chapter, we introduce the RAR greedy compression based on RAR importance index. In the RAR algorithm, the filters are pruned based on their contribution to the regression accuracy in predicting spike rates of V4 neurons. The CAR compression identifies the redundant filters in CNN structure and therefore, pruning based on CAR compression gives rise to fewer filters with diverse functionality. The RAR compression, however, is able to find the set of important CNN filters selected by each neuron because the regression accuracy is based on real neural activity.

This chapter employs both CAR and RAR compression in a transfer learning setting. Transfer learning is one of the most important techniques driving the massive success of deep learning [48]. In our setting, transfer learning consists of training a network on one

task and then transferring the weights and parameters of the trained network to perform a second target task. Our framework to build models of V4 neuron's spike rate in Chapter 3 is one example of transfer learning. We trained a CNN to classify color natural images into 1000 classes and then used the trained network to extract features from gray-scale images in neuroscience experiment and predict activity of V4 neurons. In this chapter, similar to Chapter 3, we will study transfer learning in the context of predictive models of neurons in the visual cortex [7]. First, we consider the ability of CAR-compressed networks to predict V4 neuron's activity. In these CAR-compressed CNNs, the pre-trained CNN filters are pruned based on their contributions to the classification accuracy. Then, we consider the RAR-compressed CNN-based models of V4 neurons. In these models, we first transfer the full uncompressed pre-trained network to extract features from the experiment images. Then, we prune filters based on the the regression accuracy of V4 neuron's spike rate.

Similar to Chapter 3, we study the models for neurons in visual area V4 which is poorly understood [56]. Processing of visual information by the brain is known to have hierarchical organization [54]. The input signal from retina flows from early visual cortex areas such as the V1 to intermediate layers such as the V4 and then ends with higher level visual processing in areas such as the IT area. The future research includes to extend similar modeling approach (based on compressed networks) to other visual layers. Finding the simplest accurate model for neurons in other layers uncovers an interpretable functionality for these areas. Recent lines of research in computational neuroscience have focused on building predictive models of V1 and IT areas. Building compressed and simple accurate models could further improve the current understanding of the functionality of neurons in these regions.

## 4.2 CAR-compressed predictive models of single neurons

CAR compression is a powerful tool to prune the redundant filters from CNNs. We employ CAR-compressed CNNs and a linear regression model to build predictive models for V4 neurons. Similar to the pipeline described in Chapter 3, transfer learning is used to build the models. We use the pre-trained AlexNet network trained for image classification task on the ImageNet dataset. We limit our study in this chapter to the second layer of AlexNet. The reason for this choice has been discussed in Chapter 3. The models for V4 neurons from second layer of AlexNet has one the highest average accuracies across 71 V4 neurons among all other structures (Average Correlation Coefficient of 0.53). Additionally, it has the most similar DeepTune images to the consensus DeepTune (Chapter 3). We study the CAR-compressed AlexNet in layer 2 with 9 different compression rates of 10%,

20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90%. Similar to Chapter 3, the first two layers of the network are used to extract features from images in neuroscience experiment. A ridge linear regression is used to predict V4 neuron spike rates from the extracted features.

Figure 4.1.A compares the prediction performance of the full AlexNet to the compressed networks with 9 different compression ratios. To evaluate the prediction performance, correlation coefficient between measured neural response and predicted response using CNN features are computed for the held-out test set. The test set has never been used in the training stage. The red dashed line in Figure 4.1 shows the average correlation coefficient over 71 neurons. To study effect of compression on this transfer learning task, we have pruned filters in second layer of AlexNet using greedy filter-based compression and predicted the response of neurons using compressed network. The first layer of the network is not compressed. The average correlation coefficient for networks with various compression ratios are shown in Figure 4.1.A. Comparing the average correlation of coefficients, we observe that removing half of the filters in AlexNet does not affect the prediction performance of the V4 neurons' models. Removing 90% of the filters causes a 2% reduction in the average correlation coefficient (relative 5% to uncompressed network accuracy). Figure 4.1.B illustrates scatterplots of correlation coefficients for compressed against uncompressed networks. Each point in these scatterplots corresponds to one V4 neuron. Five scatterplots for compression ratios of 10%, 30%, 50%, 70%, and 90% are shown. The reduction in prediction accuracy is negligible for the first four plots. For the 90% compression ratio plot, the accuracy is slightly lower for almost all of the neurons (68 out of 71 neurons), however, models with the 90% compression rate have only 25 filters in the second layer. Therefore, these models are much simpler compared to the original uncompressed ones with 256 filters in the second layer. It is worth noting that these 25 CAR-selected filters are universally predictive of all 71 neurons in the population of V4 neurons. In Section 4.4 we study these 25 filters in more detail and show that they are selective to a diverse set of patterns in images. This property allows the corresponding neuron models to highly accurately predict the spike rates in V4 neurons.

## 4.3 DeepTune analysis of CAR-compressed models

For each of the accurate CAR-compressed models of V4 neurons, the smooth DeepTune visualization sheds light on the pattern selectivity of that neuron. Studying the DeepTune images of the compressed models has two advantages: First, the compressed model has a fewer number of filters and therefore it is easier to interpret the patterns. Second, studying the set of DeepTune images from the compressed V4 models facilitates tools to study stability of visualization. That is, similar to the arguments in Chapter 3, the

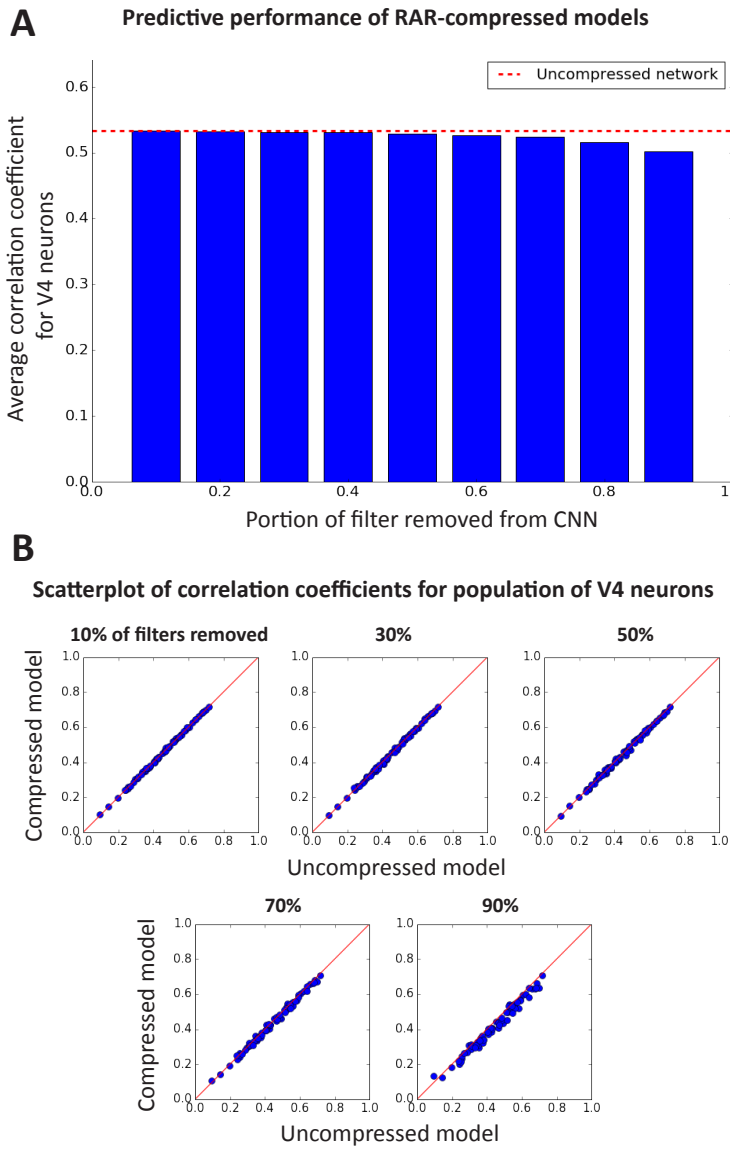


Figure 4.1: **Performance of CAR-compressed models in predicting spike rates of neurons in visal area V4. A.** Average correlation coefficient over 71 neurons for compressed network with various compression ratios. The model is based AlexNet layer 2 with Ridge regression. The red dashed line shows the average correlation coefficient for the uncompressed network. **B.** Scatterplots of correlation coefficients for compressed against uncompressed networks. Each point in these scatterplots corresponds to one V4 neuron. Five scatterplots for compression ratios of 10%, 30%, 50%, 70%, and 90% are shown.

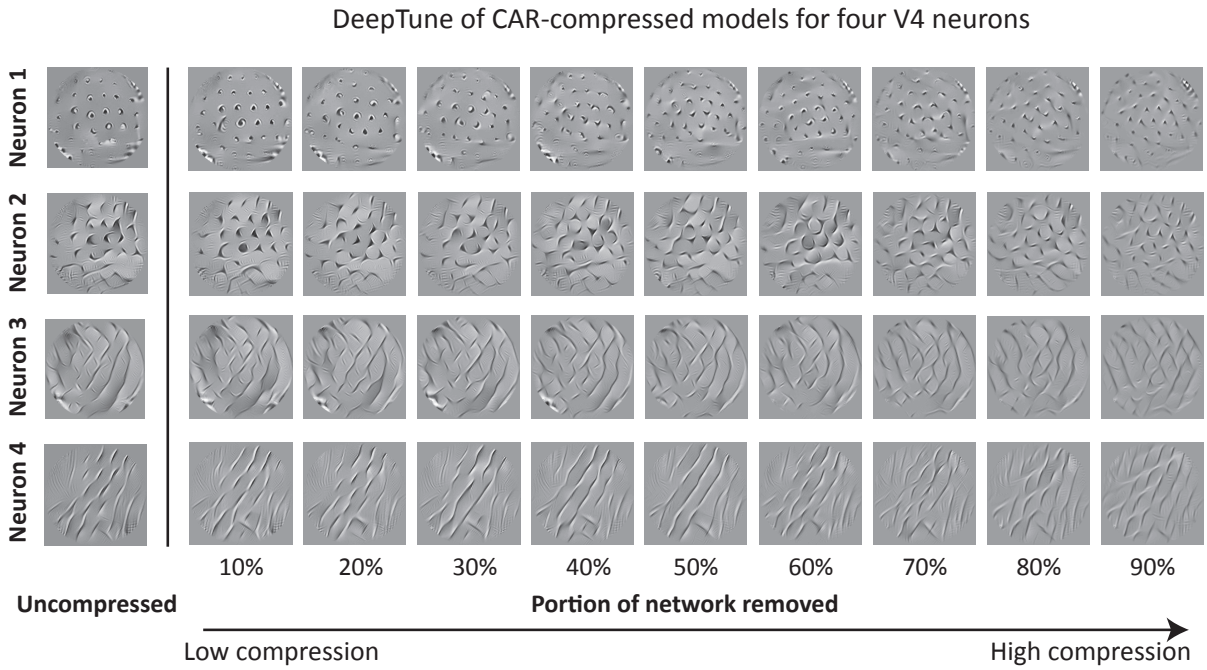


Figure 4.2: **DeepTune visualization of pattern selectivity for CAR-compressed models.** The DeepTune images for four neurons and 9 different compression rates are shown. The compressed models have stable patterns compared to uncompressed models up to a 90% compression rate.

stable patterns among Deeptune images of the compressed models with high predictive accuracy should be interpreted as V4 neuron's tuning. Here, the uncompressed model achieves a correlation coefficient of 0.53 between measured and predicted spike rates. The compressed models up to 90% compression rate achieve no less than 5% lower accuracy compared to the uncompressed model. Therefore, we limit the compression rate to to 90% in the DeepTune analysis.

Figure 4.2 illustrates the DeepTune images from compressed models with different compression rates for four neurons. For Neuron 1, the blob-like patterns appears in all compression levels up to 90%. Similarly, all compressed models for Neuron 2 lead into corner-like textures. For higher compression rates, the corners has finer edges. This happens because fewer filters (25 filters when compression rate is 90%) are involved in the modeling and visualization. The fewer filters involved in visualization reduces the redundant and irrelevant filters and therefore the DeepTune converges to sharper and finer patterns. The curvature contours and the diagonal lines are consistent among compressed models for Neurons 3 and 4, respectively. The compressed models have stable patterns

compared to uncompressed models for all of the neurons in the population and up to 90% compression rate. However, compressing the CNN with higher compression rates reduces the accuracy and therefore the corresponding DeepTune image is not reliable.

## 4.4 Visualization of CAR-important filters

In Section 4.2, if we remove 90% of filters in the second layer of AlexNet based on the CAR importance index, the prediction accuracy for V4 neurons drops from a correlation of 0.53 to 0.51, a reduction of only 0.02. This compressed CNN uses only the 25 remaining filters to predict the activities of V4 neurons. These 25 filters are visualized in Figure 4.3. Similar to Chapters 2 and 3, we show the 9 top image patches from the ImageNet training set with the highest filter responses to visualize a convolutional filter from a CNN. These 9 top image patches are representative of what this convolutional filter is computing [74, 72]. The filters are ordered based on their CAR importance index, with filter 1 being the most important filter identified by the CAR algorithm. Inspecting the visualization of these filters, we observe that the filters span a diverse set of visual patterns. The patterns include edges in different directions (filters number 5, 10, 11, 18, 22), curvatures with different orientations (filters number 12, 13, 16, 20, 24), blob-like patterns (filter number 9), textures (filters number 1, 6, 8, 23), corner-like textures (Filter number 3), and complex patterns (rest of the filters). This is consistent with our findings in Chapter 2. The diversity of the patterns in compressed network similar to uncompressed one allows the CNN-based models for V4 neurons to maintain their high accuracy even with the compression rate of 90%.

## 4.5 RAR compression

Similar to the CAR (Classification Accuracy Reduction) index introduced in Chapter 2, we define a new index based on Regression Accuracy Reduction or RAR. The RAR index quantifies the reduction in regression accuracy, when the filter in the convolutional network is pruned. In this chapter, the accuracy of regression is defined as the Pearson correlation coefficient between predicted and measured neural activity. The main reason for this choice of accuracy, as opposed to other measures such as mean squared error (MSE), is that we study RAR in the context of spike rates of neurons. Correlation coefficient is often used to evaluate accuracy in neuroscience domain because the activity of neurons in the rest state differs, and therefore the mean value and the raw magnitude should not affect the accuracy [11]. Mathematically, the regression accuracy in this work is defined as:

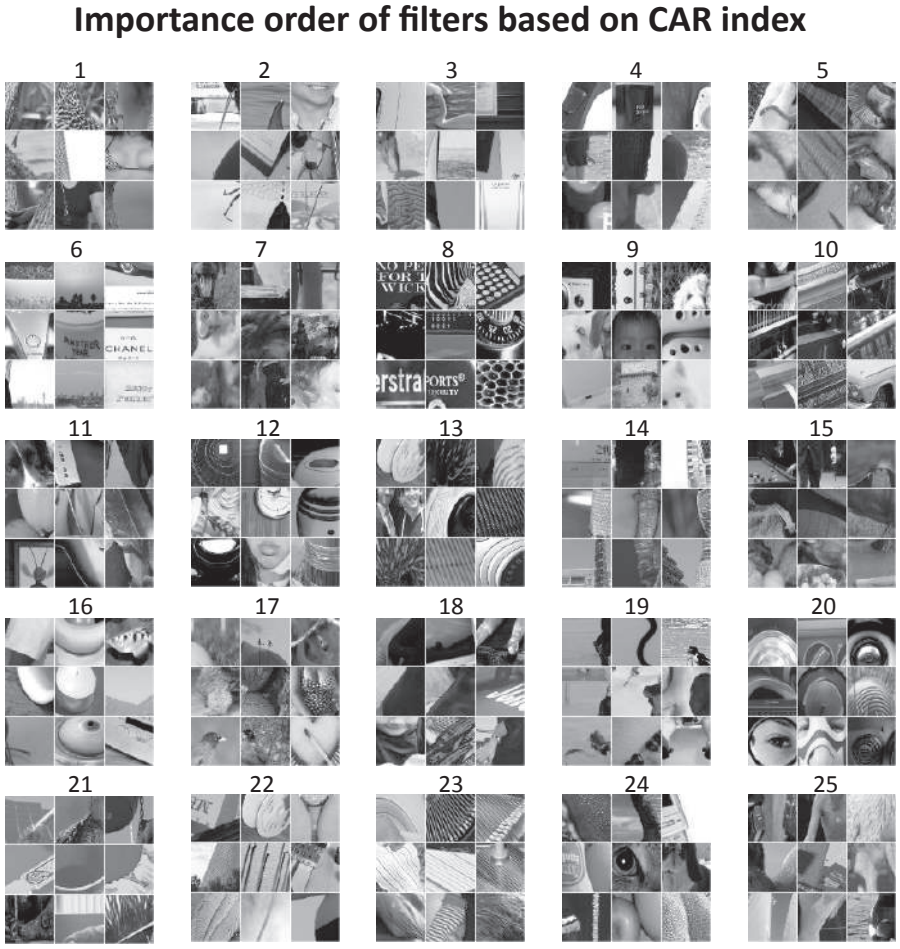


Figure 4.3: **Visualization of filters in the second layer of AlexNet with the highest CAR index.** To visualize a convolutional filter from a CNN, we show the 9 top image patches from the ImageNet training set that has the highest filter responses. The filters are ordered based on their CAR importance index.

$$Acc_{reg} = \frac{\sum_{i=1}^{N}(x^i_{pred} - \bar{x}_{pred})(x^i_{measured} - \bar{x}_{measured})}{\sqrt{\sum_{i=1}^{N}(x^i_{pred} - \bar{x}_{pred})^2}\sqrt{\sum_{i=1}^{N}(x^i_{measured} - \bar{x}_{measured})^2}}$$

The RAR index for the $i$th filter in layer $L$ of the network is defined as:

$$RAR(i, L) = Acc_{reg}(\mathcal{N}) - Acc_{reg}(\mathcal{N}(-i, L)),$$

**Algorithm 2** RAR compression

**Input:** Weights in CNN, target layer $L$ with $n_L$ filters, target compression ratio $r_{target}$

Set $n_{iter} = n_L$ and $r_{iter} = 1$
**while** $r_{iter} < r_{target}$ **do**
  **for** $i = 1$ **to** $n_L$ **do**
    Compute $RAR(i, L)$, importance index for filter $i$ in layer $L$
  **end for**
  Remove the least important filter, $\arg\min_i RAR(i, L)$
  Update compression rate, $r_{iter} = n_L / n_{iter}$
**end while**

where similar to notation in Chapter 2, network $\mathcal{N}(-i, L)$ is network $\mathcal{N}$ except that filter $i$ from layer $L$, together with all of its connections to the next layer, are removed from the network.

## 4.6 Greedy pruning algorithm based on RAR importance index

The compression of a convolutional neural network based on RAR index is similar to the CAR-based compression except that the classification accuracy is replaced with regression accuracy or correlation coefficient. In our RAR structural (or filter pruning) compression algorithm, the filter with the smallest effect on the regression accuracy gets pruned in each iteration. Similar to Chapter 2, the network can be retrained in each iteration and after pruning a filter. This process is regarded as *fine tuning* in this chapter. Algorithm 2 shows the pseudo code of our RAR greedy structural compression algorithm. In this pseudo code, $n_{iter}$ and $r_{iter}$ are, respectively, the number of remaining filters and compression ratio in the current iteration.

Note that unlike CAR compression, RAR compression is specific for each V4 neuron. Different redundant filters get pruned for different V4 neurons. Each neuron tends to keep the filters that are more relevant to its pattern selectivity. This is not the case for the CAR compression where the pruning is based on classification accuracy on ImageNet dataset and therefore, the order of pruning does not depend to the V4 neuron under study.

Similar to CAR, one can also compress based on the variants of the RAR algorithm. One possibility is to avoid greedy process and remove several filters with lowest importance indices in one pass. This compression is faster, but the performance of the compressed network is worse than Algorithm 1 in the examples we tried. The greedy process with

fine-tuning at each iteration seems to allow for a better data and network adaptation and improves compression performance. In each iteration of the greedy process, we first compute the RAR index for each filter. Then, the least important filter gets pruned. In the next iteration, we update all the importance indices using the new structure. This allows the algorithm to adapt to the new structure gradually and improves the classification accuracy.

## 4.7 RAR-compressed predictive models of single neurons

In this Section, we study the RAR-compressed CNN-based models for V4 neurons. For each neuron among the 71 V4 neurons, we first build a CNN-based model using second layer of AlexNet. Then we perform the RAR compression each individual model. That is, we remove filters in CNN based on contribution of that filter to the final regression accuracy. The filters with the lowest contribution are pruned in each iteration of RAR for each V4 neuron. Note that unlike the CAR algorithm, the RAR compression should be performed for each V4 neuron, individually. Therefore, for each neuron, a different set of filters are maintained in the network after compression.

Figure 4.4.A shows the average correlation coefficient between predicted and measured neuron spike rate across 71 neurons. The dashed line specifies the average correlation coefficient for the full uncompressed network (AlexNet layer 2 and ridge regression). The bar plots illustrates the average correlation coefficients for the RAR-compressed networks with 9 different compression rates. All the correlation coefficients are computed on the held-out test set. Similar to CAR compression, the accuracy for compressed networks up to 50% compression rate is similar to the original network. However, contrary to CAR compression, the average accuracy for compressed networks between 50% and 90% compression rates are slightly higher compared to the full network. The highest average correlation coefficient is achieved by the 80% compressed models with 0.03 higher correlation coefficient compared to the original network, though the accuracy drops for higher than 90% compression rates. The 90% RAR-compressed model contains 25 filters from the second layer of AlexNet, providing a simple accurate, CNN-inspired model of V4 neurons.

To further study the models for V4 neurons, we investigated the visualization of pattern selectivity of V4 neurons for these compressed models. The smooth DeepTune images from RAR-compressed models with 9 different compression rates are illustrated in Figure 4.4.B. Similar to CAR DeepTune images, the RAR compressed models for Neuron 1 have blob-like patterns. These patterns are consistent for the compression rates up to 90%. The

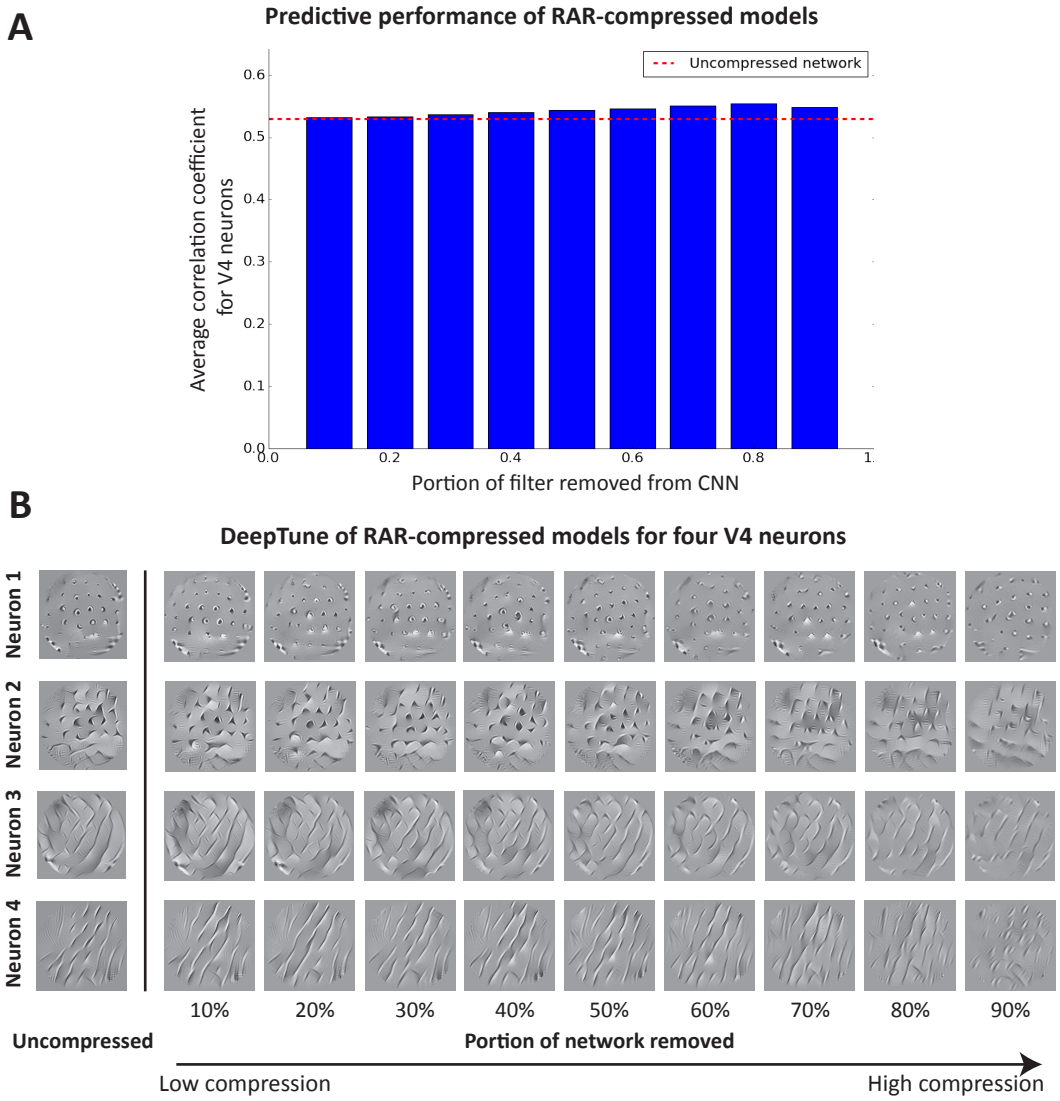


Figure 4.4: **Performance of RAR-compressed networks in predicting spike rates of neurons in visual area V4. A.** Average correlation coefficient over 71 neurons for RAR compressed networks with various compression rates. The model is based on AlexNet layer 2 with Ridge regresion. The red dashed line shows the average correlation coefficient for the uncompressed network. **B.** DeepTune visualization of pattern selectivity for RAR-compressed models. The DeepTune images for four neurons and 9 different compression rate are shown. The compressed models have stable patterns compared to uncompressed models up to 90% compression rate. For Neuron 3, the pattern is enhanced in higher compression rates.

DeepTune images for models with higher compression rates are not visualized due to their low accuracy (for the compression rate of 95%, the accuracy drops to 0.35). Therefore the models are not reliable. For Neuron 2, the compressed models with higher than 60% compression rate tend to have highly sparse patterns. This is also true for Neurons 3 and 4. For the rest of the neurons in the population, this observation remains true for close to 75% of the neurons. Having sparser patterns in DeepTune images is preferable because the patterns are easier to interpret. This is consistent with our original motivation for removing filters from CNN to have simpler models. Additionally, the RAR-compressed models enhance the visualization for some of the neurons. For Neuron 3, the DeepTune image from 90% compressed model has weaker diagonal patterns in 45 deg orientation. The prediction accuracy for the compressed model is 0.55 which is a relative 6% higher accuracy compared to uncompressed model for this neuron. Therefore, the enhanced visualization for this neuron has stronger anti-diagonal patterns (with $-45$ deg orientation) and weaker diagonal patterns (with 45 deg orientation).

Our results show that the RAR-compressed computational models for V4 models have the following advantages over the uncompressed models. First, the models identified by the RAR compression have simpler structure with far fewer filters compared to uncompressed models. These models have higher predictive accuracy of the spike rates for V4 neurons (up to 90% compressed rate for V4 neurons). Finally, the visualization of the pattern selectivity of neurons based on compressed models are stable, showing similar patterns to the uncompressed models.

## 4.8 Discussion

Both CAR and RAR compressions provide powerful tools to remove filters from CNNs. We employ these compressed CNNs to build predictive models of V4 neurons in primate's visual cortex. CAR compression identifies a universal subset of filters in CNN with diverse functionalities. These filters are basis of the models for V4 neurons. RAR compression, however, selects the subset of filters according to each V4 neuron.

Interestingly, the average prediction accuracy of RAR-compressed V4 models are higher compared to uncompressed models (up to 90% compression rate). For 95% of the neurons in the visual area V4, the accuracy of RAR-compressed models with compression ratio 80% is higher compared to the original models. However, the RAR compression requires higher computational cost compared to CAR. The greedy filter selection operation for each neuron should be repeated for each of the V4 neuron. In our case, the number of neurons (and therefore the number of repeating operation) is not high and therefore the computation is feasible. In some other applications, the computational cost might be a

more serious limitation. For example, performing RAR compression to find a sub-optimal model for predicting voxels in fMRI study is not immediately feasible because the number of voxels under study are often in the order of thousands. Future work on decreasing the computational cost of RAR compression will make them more accessible in higher dimensional problems. Parallelizing the computations in addition to employing methods to identify a subset of filters instead of one filter in each iteration are some solutions to reduce the computational cost.

# Bibliography


[1] Reza Abbasi-Asl and Bin Yu. "Interpreting Convolutional Neural Networks Through Compression". In: *NIPS 2017 symposium on Interpretable Machine Learning, arXiv: 1711.02329* (2017).

[2] Reza Abbasi-Asl and Bin Yu. "Structural Compression of Convolutional Neural Networks Based on Greedy Filter Pruning". In: *arXiv preprint arXiv:1705.07356* (2017).

[3] John Allman, Francis Miezin, and EveLynn McGuinness. "Stimulus specific responses from beyond the classical receptive field: neurophysiological mechanisms for local-global comparisons in visual neurons". In: *Annual review of neuroscience* 8.1 (1985), pp. 407–430.

[4] Fabrice Arcizet, Christophe Jouffrais, and Pascal Girard. "Natural textures classification in area V4 of the macaque monkey". In: *Experimental brain research* 189.1 (2008), pp. 109–120.

[5] Francis R Bach. "Bolasso: model consistent lasso estimation through the bootstrap". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 33–40.

[6] Steve Branson et al. "Bird species categorization using pose normalized deep convolutional nets". In: *arXiv preprint arXiv:1406.2952* (2014).

[7] Charles F Cadieu et al. "Deep neural networks rival the representation of primate IT cortex for core visual object recognition". In: *PLoS Comput Biol* 10.12 (2014), e1003963.

[8] Matteo Carandini et al. "Do we know what the early visual system does?" In: *Journal of Neuroscience* 25.46 (2005), pp. 10577–10597.

[9] Eric T Carlson et al. "A sparse object coding scheme in area V4". In: *Current Biology* 21.4 (2011), pp. 288–293.

[10] Supriyo Chakraborty et al. "Interpretability of deep learning models: a survey of results". In: *Interpretability of deep learning models: a survey of results*. 2017.

[11] Marlene R Cohen and Adam Kohn. "Measuring and interpreting neuronal correlations". In: *Nature neuroscience* 14.7 (2011), p. 811.

[12] Stephen V David, Benjamin Y Hayden, and Jack L Gallant. "Spectral receptive field properties explain shape selectivity in area V4". In: *Journal of neurophysiology* 96.6 (2006), pp. 3492–3505.

[13] Stephen V David, William E Vinje, and Jack L Gallant. "Natural stimulus statistics alter the receptive field structure of v1 neurons". In: *Journal of Neuroscience* 24.31 (2004), pp. 6991–7006.

[14] Russell L De Valois, E William Yund, and Norva Hepler. "The orientation and direction selectivity of cells in macaque visual cortex". In: *Vision research* 22.5 (1982), pp. 531–544.

[15] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.

[16] Robert Desimone and Stanley J Schein. "Visual properties of neurons in area V4 of the macaque: sensitivity to stimulus form". In: *Journal of neurophysiology* 57.3 (1987), pp. 835–868.

[17] Finale Doshi-Velez and Been Kim. "A roadmap for a rigorous science of interpretability". In: *arXiv preprint arXiv:1702.08608* (2017).

[18] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: (2017).

[19] Jack L Gallant, Jochen Braun, and David C Van Essen. "Selectivity for polar, hyperbolic, and Cartesian gratings in macaque visual cortex". In: *Science* 259.5091 (1993), pp. 100–103.

[20] Jack L Gallant et al. "Neural responses to polar, hyperbolic, and Cartesian gratings in area V4 of the macaque monkey". In: *Journal of neurophysiology* 76.4 (1996), pp. 2718–2739.

[21] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[22] Riccardo Guidotti et al. "A Survey Of Methods For Explaining Black Box Models". In: *arXiv preprint arXiv:1802.01933* (2018).

[23] Song Han, Huizi Mao, and William J Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding". In: *arXiv preprint arXiv:1510.00149* (2015).

[24] Babak Hassibi, David G Stork, and Gregory J Wolff. "Optimal brain surgeon and general network pruning". In: *Neural Networks, 1993., IEEE International Conference on*. IEEE. 1993, pp. 293–299.

[25] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[26] Tianxing He et al. "Reshaping deep neural network for fast decoding by node-pruning". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 245–249.

[27] David H Hubel and Torsten N Wiesel. "Receptive fields and functional architecture of monkey striate cortex". In: *The Journal of physiology* 195.1 (1968), pp. 215–243.

[28] Forrest N Iandola et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size". In: *arXiv preprint arXiv:1602.07360* (2016).

[29] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[30] Yong-Deok Kim et al. "Compression of deep convolutional neural networks for fast and low power mobile applications". In: *international Conference on Learning Representations*. 2016.

[31] Eucaly Kobatake and Keiji Tanaka. "Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex". In: *Journal of neurophysiology* 71.3 (1994), pp. 856–867.

[32] Hideki Kondo and Hidehiko Komatsu. "Suppression on neuronal responses by a metacontrast masking stimulus in monkey V4". In: *Neuroscience research* 36.1 (2000), pp. 27–33.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[34] B Boser Le Cun et al. "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems*. Citeseer. 1990.

[35] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[36] Yann LeCun et al. "Optimal brain damage." In: *NIPS*. Vol. 2. 1989, pp. 598–605.

[37] Hao Li et al. "Pruning Filters for Efficient ConvNets". In: *arXiv preprint arXiv:1608.08710* (2016).

[38] Hao Li et al. "Pruning filters for efficient convnets". In: *arXiv preprint arXiv:1608.08710* (2017).

[39] Zachary C Lipton. "The mythos of model interpretability". In: *arXiv preprint arXiv:1606.03490* (2016).

[40] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[41] Scott Lundberg and Su-In Lee. "An unexpected unity among methods for interpreting model predictions". In: *arXiv preprint arXiv:1611.07478* (2016).

[42] Aravindh Mahendran and Andrea Vedaldi. "Understanding deep image representations by inverting them". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5188–5196.

[43] William H Merigan. "Cortical area V4 is critical for certain texture discriminations, but this effect is not dependent on attention". In: *Visual neuroscience* 17.6 (2000), pp. 949–958.

[44] Pavlo Molchanov et al. "Pruning Convolutional Neural Networks for Resource Efficient Inference". In: (2017).

[45] Anirvan S Nandy et al. "The fine structure of shape tuning in area V4". In: *Neuron* 78.6 (2013), pp. 1102–1115.

[46] Shinji Nishimoto et al. "Reconstructing visual experiences from brain activity evoked by natural movies". In: *Current Biology* 21.19 (2011), pp. 1641–1646.

[47] Gouki Okazawa, Satohiro Tajima, and Hidehiko Komatsu. "Image statistics underlying natural texture selectivity of neurons in macaque V4". In: *Proceedings of the National Academy of Sciences* 112.4 (2015), E351–E360.

[48] Maxime Oquab et al. "Learning and transferring mid-level image representations using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1717–1724.

[49] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[50] Anitha Pasupathy and Charles E Connor. "Population coding of shape in area V4". In: *Nature neuroscience* 5.12 (2002), p. 1332.

[51] Anitha Pasupathy and Charles E Connor. "Responses to contour features in macaque area V4". In: *Journal of Neurophysiology* 82.5 (1999), pp. 2490–2502.

[52] Ryan Prenger et al. "Nonlinear V1 responses to natural scenes revealed by neural network analysis". In: *Neural Networks* 17.5 (2004), pp. 663–679.

[53] Russell Reed. "Pruning algorithms-a survey". In: *IEEE transactions on Neural Networks* 4.5 (1993), pp. 740–747.

[54] Maximilian Riesenhuber and Tomaso Poggio. "Hierarchical models of object recognition in cortex". In: *Nature neuroscience* 2.11 (1999), pp. 1019–1025.

[55] J Cooper Roddey, B Girish, and John P Miller. "Assessing the performance of neural encoding models in the presence of noise". In: *Journal of computational neuroscience* 8.2 (2000), pp. 95–112.

[56] Anna W Roe et al. "Toward a unified theory of visual area V4". In: *Neuron* 74.1 (2012), pp. 12–29.

[57] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: Nonlinear Phenomena* 60.1-4 (1992), pp. 259–268.

[58] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[59] Stanley J Schein and Robert Desimone. "Spectral properties of V4 neurons in the macaque". In: *Journal of Neuroscience* 10.10 (1990), pp. 3369–3389.

[60] Oliver Schoppe et al. "Measuring the performance of neural models". In: *Frontiers in computational neuroscience* 10 (2016).

[61] Ali Sharif Razavian et al. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 806–813.

[62] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[63] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[64] Robert Tibshirani. "Regression Selection and Shrinkage via the Lasso". In: *Journal of the Royal Statistical Society B* 58 (1994), pp. 267–288. ISSN: 00359246. DOI: 10.2307/2346178. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.7574.

[65] Jon Touryan and James A Mazer. "Linear and non-linear properties of feature selectivity in V4 neurons". In: *Frontiers in systems neuroscience* 9 (2015).

[66] Ben D B Willmore, Ryan J Prenger, and Jack L Gallant. "Neural representation of natural images in visual area V2". In: *The Journal of neuroscience* 30.6 (2010), pp. 2102–2114.

[67] Ben DB Willmore, James A Mazer, and Jack L Gallant. "Sparse coding in striate and extrastriate visual cortex". In: *Journal of neurophysiology* 105.6 (2011), pp. 2907–2919.

[68] Jiaxiang Wu et al. "Quantized convolutional neural networks for mobile devices". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4820–4828.

[69] Michael C-K Wu, Stephen V David, and Jack L Gallant. "Complete functional characterization of sensory neurons by system identification". In: *Annu. Rev. Neurosci.* 29 (2006), pp. 477–505.

[70] Daniel L K Yamins and James J DiCarlo. "Using goal-driven deep learning models to understand sensory cortex". In: *Nature Neuroscience* 19.3 (2016), pp. 356–365.

[71] Daniel L K Yamins et al. "Performance-optimized hierarchical models predict neural responses in higher visual cortex". In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8619–8624.

[72] Jason Yosinski et al. "Understanding neural networks through deep visualization". In: *arXiv preprint arXiv:1506.06579* (2015).

[73] Bin Yu. "Stability". In: *Bernoulli* 19.4 (2013), pp. 1484–1500.

[74] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[75] Peng Zhao and Bin Yu. "On model selection consistency of Lasso". In: *The Journal of Machine Learning Research* 7 (2006), pp. 2541–2563.