Modeling the Early Visual System

Nicholas Lanning
University of Kentucky, lanningn17@gmail.com
Author ORCID Identifier:
https://orcid.org/0009-0006-5764-0449
Digital Object Identifier: https://doi.org/10.13023/etd.2023.490

MODELING THE EARLY VISUAL SYSTEM

THESIS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering in the College of Engineering at the University of Kentucky

By
Nicholas Lanning
Lexington, Kentucky

Director: Dr. Luis Sanchez Giraldo, Professor of The Department of Electrical and Computer Engineering
Lexington, Kentucky
2023

ABSTRACT OF THESIS

MODELING THE EARLY VISUAL SYSTEM

There are two encoding schema present in simple cells in the early visual system of vertebrates: the retinal simple cells activate highly when the receptive field contains a center surround stimulus, while the primary visual cortex's (V1) simple cells activate highly when the receptive field contains visual edges. Work has been done in the past to enforce constraints on visual machine learning such that the retinal or V1 encoding is learned, but this work is often done to emulate retinal and V1 encoding in a vacuum. Recent work using convolutional neural networks focuses on anatomical constraints along with a supervised objective for training the network to explain the emergent representations of retina and V1 in vertebrates. The model dismisses observations made by other models of retinal processing where robustness to noise and coding efficiency are considered. Moreover, the use of a convolutional architecture explicitly enforce spatial equivariance in the features, which can limit the emergence of other relevant features. Here, we explore a more flexible model. We propose the EVSNet, a fully-connected neural network which learns retinal and V1 features. To analyze the representations learned with this network, we propose a measure called the orientedness to quantitatively discern expected retinal features from expected V1 features.

KEYWORDS: Early Visual System, Representation Learning, Information Bottleneck

Nicholas Lanning

December 14, 2023

MODELING THE EARLY VISUAL SYSTEM

By
Nicholas Lanning

Dr. Luis Sanchez Giraldo
Director of Thesis

Dr. Daniel Lau
Director of Graduate Studies

December 14, 2023
Date

For Yancey

ACKNOWLEDGMENTS

I would like thank Dr. Luis Gonzalo Sanchez Giraldo for his continuous support and guidance during my time working in his lab. I also thank Oscar Skean, Keider Hoyos, and Santiago Posso for their understanding, wisdom, and camaraderie. I finally would like to thank my family for their support and faith in me as I ventured through this chapter of my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

The act of seeing is something which may easily be taken for granted, but the human visual system contains many complex parts, some of which are not fully understood. As such, studying and modeling aspects of human and vertebrate visual systems has been a much-pursued field of study.

Early work in the 1950's and 1960's was devoted to experimentally learning the response of simple cells in the retina and primary visual cortex (V1) of cats by applying simple light stimuli and observing the response [33, 26, 27]. Additional physiological experimentation was done with the early visual system of old world primates [28, 17], which observed similar retinal and cortical simple cell activations to stimuli as was observed with cats. Using human perceptual experiments, the goals and architecture of the early primate visual system were correlated with the human visual system [37].

Occurring around the same time as the physiological cat studies, efforts were focused on theorizing the informational objectives of vision and its components [5, 3, 35]. One of the key ideas presented was the redundancy reduction theory, which assumed that the goal of the early visual system was to transmit information while reducing the redundancy of the sent information, meaning that the early visual system would be efficiently encoding visual information. Later during the 1980's and 1990's interest in information theory as it pertains to sensory processing was revitalized [34, 54, 20, 2, 1, 7, 58, 21, 45], and moving from the late 1990's into the 2000's and 2010's working with visual machine learning models became more accessible. [45, 59, 18, 50].

The success of modern computer vision systems in complex tasks like visual object recognition and segmentation has motivated investigating the similarities and differences between these artificial systems and their biological counterparts [32]. On the other hand, understanding the mechanisms that biological systems employ can serve as inspiration to improve computer vision systems [15]. Representation learning is a machine learning method by which we try to learn a set of elements which can used to represent some input data. Indeed, work has been done previously to learn representations which match physiological retinal and V1 encodings, but this work largely focuses on developing these representations in a vacuum, not as part of a visual system. In addition, the work that has been done developing the two representations in tandem assumes spatial equivariance in the learned features by using a convolutional neural network, and makes some deviations from most other early visual representation learning in terms of dataset and training task. To address these issues, we propose an Early Visual System Network (EVSNet) which:

- learns both retinal and V1 representations in tandem,
- uses fully-connected layers to validate whether or not spatial equivariance can be assumed of the learned features,

- utilizes the popular van Hateren Natural Image Dataset [57] and an unsupervised learning task.

We also use several methods to analyze the representations learned with this network, including a measure we propose called the orientedness measure. This measure helps to quantitatively discern expected retinal features from expected V1 features.

1.1 Outline

Chapter 1 provides a brief overview of the history of research leading up to the state of the art in modeling the early visual system of vertebrates, as well as an outline and the objectives of this thesis. Chapter 2 explains principles regarding neural networks and how they apply to biological visual systems, unsupervised learning and autoencoders, residual networks, and how to visualize the receptive fields learned by the network. Familiarity with this information will be vital in understanding the methods from prior research and the methods we present. Chapter 3 will detail some of the important experimentation concerning early visual system learning in order to further contextualize the state of the art. Chapter 4 shows the specific techniques we use and the considerations we make given the existing body of work. Chapter 5 contains visualizations of the learned representations for various parameter sweeps, as well as some other pertinent analyses of these models. Chapter 6 will discuss implications of the presented results and key takeaways. Finally, Chapter 7 summarizes the ideas for future work which could be used to develop further on this research.

1.2 Objectives

1.2.1 Main Objective

Construct a model for the early visual system via representation learning.

1.2.2 Specific Objectives

- Construct a model using fully-connected layers in an autoencoder network which models the retinal and V1 encoding in tandem.
- Utilize biologically accurate constraints on learning in order to encourage the expected retinal and V1 representations.
- Gain understanding of how adjusting network hyperparameters can affect the learned representation of the retinal and V1 layers.
- Propose a method for quantitatively differentiating different receptive field shapes.
- Compare our conditions for learning receptive fields to existing work.

Chapter 2 Background

2.1 Receptive Fields

In the visual systems of vertebrates, two of the first components of visual sensing and processing are the retina and V1 [25]. In the retina, many photoreceptors at nearby locations in space report their sensed information to retinal ganglion cells, which aggregate this information to provide a single activation for a large group of photoreceptors and their associated portion of the visual field. The retinal ganglion cells activate highly when they sense a pattern in their portion of the visual field, known as a receptive field. It has been found via physiological experimentation with vertebrates that these retinal ganglion cells have the highest activations when the visual field contains what is called a “center surround” receptive field [33], shown in Figure 2.1. In contrast, neurons in V1 have been experimentally shown to activate highly when the visual receptive field contains what is known as an oriented Gabor receptive field [26], shown in Figure 2.2.

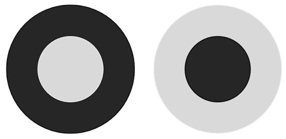


Figure 2.1: On-center and off-center center surround receptive fields. Light and dark regions correspond to light or dark regions in the visual field; for example, a neuron with the off-center receptive field at right would activate highly for a visual stimulus with a dark center and bright surround.

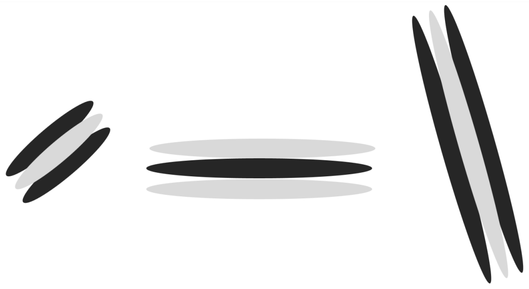


Figure 2.2: Oriented Gabor receptive fields. Gabors receptive fields may have a light or dark center, where light portions mean that the neuron responds to light in that region, while dark regions mean that the neuron responds to the lack of light in that region.

2.2 Supervised vs. Unsupervised Learning

Typically, retinal and V1 receptive fields are modeled using unsupervised learning techniques. Unsupervised learning can be described as "learning without a teacher". The model should capture the structure in the data from regularities. On the other hand, as we move up the hierarchy, supervised learning models where the goal of learning is to find a mapping from inputs to desired behaviors have been more successful [56]. In computer vision, typical uses of supervised learning include object classification or object detection. The learning is "supervised" because training is done with data where we know the correct answer and can therefore punish incorrect answers or reward correct answers. For example, in digit classification a network may be trained with image data that is paired with the label of a digit present in the image, and for object detection the data is paired with a boolean mask which indicates whether or not the object is present. In contrast to this, unsupervised learning methods for computer vision do not operate with a notion of what is correct or incorrect. Some unsupervised methods, such as principal component analysis (PCA) and independent component analysis (ICA), utilize optimization of statistics of the input data in order to learn underlying components of the data and form a representation. Often times, as in the case of PCA, this learned representation corresponds to a lower dimensional linear subspace of the input data, meaning the input data or other signals similar to the input data can be represented in a more compact manner. Another important unsupervised method in the field of representation learning is the autoencoder.

2.3 Autoencoders

The idea of finding a compact representation of the input data can be generalized by employing transformations of the input data. To guarantee that the information about the input data is preserved, such transformations should reversible. This can be accomplished by finding a transformation from the representation space back to the input space. This paradigm, which will be described in more detail below, is known as an autoencoder.

2.3.1 Architecture

An autoencoder is a model composed of an encoder and a decoder block. An encoder is a function $\tilde{G}$ which maps input data point $\mathbf{x} \in \mathbb{R}^d$ to an encoding vector $\mathbf{y} \in \mathbb{R}^p$, while a decoder is a function $\tilde{G}^{-1}$ which maps the encoding to a reconstruction of the input data, $\hat{\mathbf{x}} \in \mathbb{R}^d$ [1]. These mapping functions can have varying degrees of complexity, from linear mappings, to nonlinear operations expressed as a hierarchy of multiple transformations. A visualization of an autoencoder network can be seen in Figure 2.3.

[1]We denote the decoder by $\tilde{G}^{-1}$ to emphasize the role of reversing the encoding. However, the decoder may not correspond to an inverse map in the strict mathematical sense as $\tilde{G}$ is not necessarily invertible

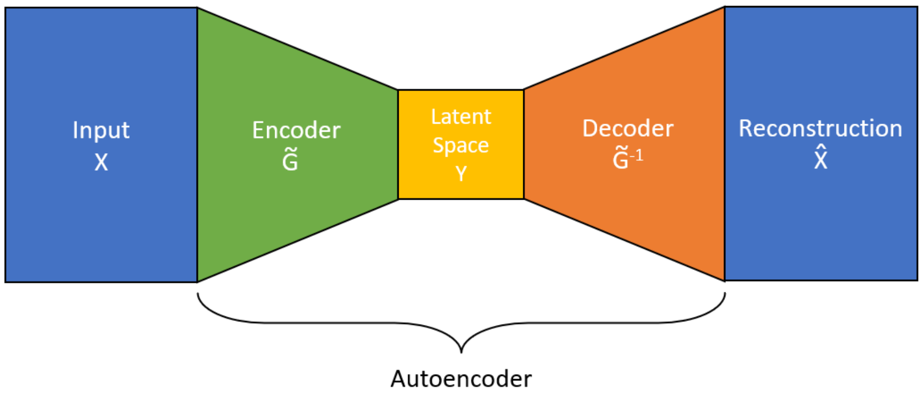


Figure 2.3: A simple autoencoder structure. The encoder produces the encoding vector, which can also be referred to as the learned representation, in the latent space. The decoder transforms the encoding vector to produce a reconstruction of the input.

At its simplest, an encoder does a linear operation on the input data using a learned weight matrix and a learned bias component. The result of this linear operation can then be passed through a nonlinear activation function, which allows the network to capture information about the input which could not be captured merely by the linear operation. This composition can be expressed as follows:

$$\mathbf{y} = g(\mathbf{W}_e\mathbf{x} + \mathbf{b}_e) \tag{2.1}$$

where $\mathbf{x}$ is the input data vector, $\mathbf{W}_e$ is the learned weight matrix, $\mathbf{b}_e$ is the learned bias vector, g is the nonlinear activation function, and $\mathbf{y}$ is the vector encoding of the input. The weight matrix, therefore, is the encoder dimensionality p (the layer width) by the input dimensionality d, and the bias is a vector with the same dimensionality as the encoder. The encoder dimensionality or layer width is the number of hidden units which can be chosen arbitrarily. Hidden units are a sort of resource which determines the complexity of the learned representation; a encoder layer with many hidden units will be able to encode more data about the input. If the encoding is a smaller dimensional space than the input, then the representation is said to be undercomplete; if it is the same size or larger, the encoding is said to be complete or overcomplete, respectively.

After the data is encoded, the encoding $\mathbf{y} \in \mathbb{R}^p$ is passed as an input to the decoder, whose job is to produce a reconstruction $\hat{\mathbf{x}}$ of the input data $\mathbf{x}$ using the encoding. In its simplest form, the decoder can be represented purely by a linear operation:

$$\hat{\mathbf{x}} = \mathbf{W}_d\mathbf{y} + \mathbf{b}_d. \tag{2.2}$$

It should be noted that $\mathbf{W}_d$ and $\mathbf{b}_d$ are the decoder weights and biases, which can be trained simultaneously but are distinct from the encoder weights and biases.

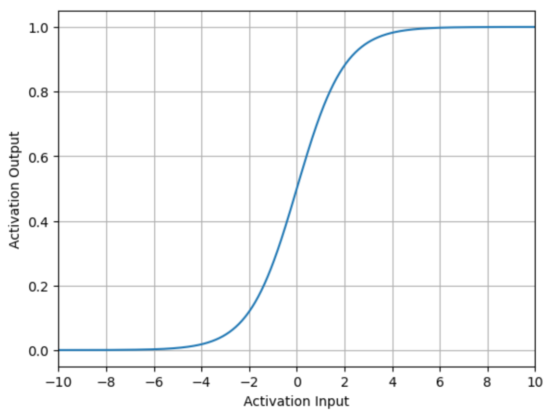


Figure 2.4: The sigmoid activation curve.

Activation Function

The most common activation functions work as point-wise nonlinearities, that is, they operate on each dimension individually. Two main types of nonlinear activation function, which will be the focus of our work, are the sigmoid and the rectified linear response. The sigmoid activation, as its name suggests, is an S-shaped curve. While there are many functions that satisfy this definition, we center our attention on the **logistic function** given by

$$y = \frac{1}{1+e^{-x}} \tag{2.3}$$

When discussing activation functions, we will use x to denote the input to the function and y to denote the output, which should be recognized as distinct from $\mathbf{x}$ and $\mathbf{y}$. Figure 2.4, shows a plot of the logistic function (2.3). As we can see the range of the function is the interval $(0, 1)$ and the domain is the entire real line. The function squeezes large values of x, either negative or positive into saturation at 0 or 1, respectively. In this sense, we can think about the outputs of the logistic function as the probability of the neural unit being active.

The second type of nonlinear activation function that we will consider is the Rectified Linear Unit (ReLU) [42]. The ReLU response is given by:

$$y = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases} \tag{2.4}$$

From Figure 2.5, we can see that the ReLU activation is essentially a max operation between 0 and x. In neural coding, a popular hypothesis is that neurons encode information about the input stimulus in what is known as firing rate [40]. Firing rate is the frequency at which action potentials occur in a neuron. The firing rate varies with the intensity of the stimulus and cannot be negative by definition. The outputs of a ReLU activation in an artificial neural network mimic this behavior.

When we talk about a learned representation of the data in autoencoders, typically we are interested in the encoder map, and the features that it captures about the input. With images for example, one way to understand the learned representation is

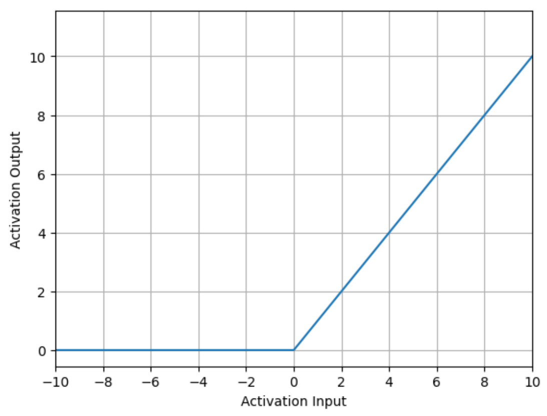


Figure 2.5: The ReLU activation curve.

to look at the inputs that contribute to the activation of each of the output dimensions of the encoder. In this sense, we can think of representing the input image as a combination of basic features in the dictionary that activate different dimensions of the encoding vector. For a given input image, the latent space activations may have some high values and some low values; these high activations are features which are very prominent in the input. For the simplest form of encoder (2.1), we can see how the information of the input can be expressed as a combination of basic features by looking at each of the rows of $\mathbf{W}_e$.

So far, we have discussed the operations that describe how the input can be transformed into the representation space (encoding vector) but we have not yet discussed how to obtain such a transformation. Intuitively, the encoder and decoder transformations must be aligned to the structure of the input data. This is accomplished by a learning process that tunes the weights and biases of the encoder and decoder. In the following section, we discuss this process in more detail.

Batch Processing

The autoencoder network operation can be represented compactly using matrix notation. A set of n inputs $\{\mathbf{x}_i\}_{i=1}^n$, or what we call a batch of size n, can be represented as a matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ with a number of rows given by the total number of input dimensions and a number of columns given by the batch size. The output of the encoder (2.1) for a batch $\mathbf{X}$, is represented by a matrix $\mathbf{Y} \in \mathbb{R}^{p\times n}$, given by:

$$\mathbf{Y} = g(\mathbf{W}_e\mathbf{X} + \mathbf{b}_e\mathbf{1}_n^T) \tag{2.5}$$

where $\mathbf{1}_n$ is an n-dimensional vector, with all entries equal to 1. The matrix $\mathbf{Y}$ is then passed as an input to the decoder, whose job is to produce a reconstruction of the input data $\mathbf{X}$ from the encoding. The linear decoding for a batch of size n is represented in matrix form as

$$\hat{\mathbf{X}} = \mathbf{W}_d\mathbf{Y} + \mathbf{b}_d\mathbf{1}_n^T. \tag{2.6}$$

2.3.2 Learning

Objective Function

The objective function, which can be arbitrarily chosen by the user, acts as a measure of how "good" a given representation is under a given criteria. For instance, in image compression, a reasonable choice would measure the reduction in size for storage and the perceptual quality of the recovered image. Conventionally, autoencoders are trained to learn a representation from which the input can be recovered up to some level of error when compared to the input data. In other words, the representation should preserve information about the input. Ideally, we would train an encoder to find a representation that maximizes the mutual information between the input and the encoded data:

$$I(X;Y) = H(Y) - H(Y|X) \tag{2.7}$$

Alternatively, we can seek to maximize $I(X;\hat{X})$. The data processing inequality states that for $X \rightarrow Y \rightarrow \hat{X}$ we have $I(X;Y) \geq I(X;\hat{X})$, as we cannot increase the mutual information between two random variables by doing more transformations, we can only maintain or reduce the mutual information. A method of note for learning a good representation is the information bottleneck, which attempts to maximize $I(X;\hat{X})$ while explicitly minimizing $I(X;Y)$ [55]. By minimizing the mutual information between the input and the latent space, the information bottleneck "squeezes" out unnecessary information, only keeping what is necessary to make a good reconstruction of the input.

Despite $I(X;\hat{X})$ being the ideal measure, computing this mutual information can be intractable since the probability distribution $P_X(\mathbf{x})$ of X is usually unknown [14]. For this reason, either a tractable lower bound on the mutual information or a surrogate for mutual information is optimized. Perhaps the most common surrogate for mutual information for real valued vector random variables is simply the mean-squared error between the input and the decoder reconstruction, given by:

$$\mathcal{L}(X,\hat{X}) = \text{MSE}(X,\hat{X}) = \mathbb{E}\left[\left\|X - \hat{X}\right\|^2\right]. \tag{2.8}$$

Given that in our case we have a set of samples $\{\mathbf{x}_i\}_{i=1}^n$, the MSE is estimated empirically as:

$$\hat{\text{MSE}}(\mathbf{X},\hat{\mathbf{X}}) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 . \tag{2.9}$$

In the case of grayscale images, $\mathbf{x}_i$ is a vector with length d given by the number of pixels. The squared norm of the error between the input image $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$ is,

$$\|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \sum_{j=1}^{d} \left((\mathbf{x})_j - (\hat{\mathbf{x}})_j\right)^2, \tag{2.10}$$

where $(\mathbf{x})_j$ is a scalar denoting the jth entry of the vector $\mathbf{x}$.

Other terms can be added to the objective function in order to emphasize other aspects that fit our definition of a "good" learned representation. For example, a common term to add is a weight regularization term. This term serves to limit the network weights from becoming too large, which may lead to unstable networks, while also preventing features from overfitting the data. Two common forms of weight regularization are the L_1 and L_2 norms:

$$\mathcal{L}_{L1reg} = \sum_{l,k} \left|(\mathbf{W})_{l,k}\right| \tag{2.11}$$

$$\mathcal{L}_{L2reg} = \sum_{l,k} (\mathbf{W})_{l,k}^2 \tag{2.12}$$

where $\mathbf{W}$ corresponds to a weight matrix that is present in any layer of a network, and l and k are indices to refer to positions in the weight matrix. When any one of these terms is incorporated into the loss, the training will not only try to minimize the error between the input and the autoencoder's reconstruction but also do it in such a way that the weights in the network have small values. Namely, $\mathcal{L}_{L1reg}$ encourages many weights to become zero which can help further interpretation of the learned weights. The $\mathcal{L}_{L2reg}$ term encourages smoothness in the weight distribution. Intuitively, the small values of the weights yield functions implemented by the network that do not change abruptly, enforcing some form of regularity on the structures learned by the autoencoder. We will use both of these norms in our objective function when training the EVSNet.

Gradients

The objective function provides a measure of how good a given configuration (a network with a given set of weight values) is. In order for the network to learn weights and biases, we search for configurations where the objective function value is high and then move the weights and biases in the direction of those configurations. To better illustrate the learning process, we will use a simple autoencoder based on (2.1) and (2.2) to compute $\hat{\mathbf{X}}$. We can treat minimizing the mean-squared error between the input and reconstruction as maximizing a lower bound on the mutual information between the input and the reconstruction [60]. Using (2.8) to represent the value we are trying to minimize, we can structure the learning of weights and biases as a minimization problem:

$$\underset{\mathbf{W}_e,\mathbf{b}_e,\mathbf{W}_d,\mathbf{b}_d}{\text{minimize}} \mathcal{L}(\mathbf{X}, \mathbf{W}_d[g\{\mathbf{W}_e\mathbf{X} + \mathbf{b}_e\mathbf{1}_n^T\}] + \mathbf{b}_d\mathbf{1}_n^T) \tag{2.13}$$

If the objective function is differentiable with respect to the parameters, we can compute the gradients of the loss with respect to the encoder and decoder parameters, $\mathbf{W}_e$, $\mathbf{b}_e$, $\mathbf{W}_d$, and $\mathbf{b}_d$. Since the gradient points to the direction where the function increases the most, we can then make small adjustments to these parameters in the opposite direction of the gradient which minimize the loss, and after many steps determined by the gradient we can reach a configuration that minimizes the objective function [12].

There is an efficient way to structure finding the gradients of the loss with respect to each of the network parameters known as backpropagation [52]. We start with the expression for the loss, and then we use the chain rule to move backwards through the network until we reach the desired parameter. For example, to find the partial derivative of the loss with respect to the decoder weights in our example two layer autoencoder, we would do:

$$\frac{\partial \mathcal{L}}{\partial \left(\mathbf{W}_d\right)_{\ell,k}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d} \frac{\partial \mathcal{L}}{\partial \left(\hat{\mathbf{X}}\right)_{j,i}} \frac{\partial \left(\hat{\mathbf{X}}\right)_{j,i}}{\partial \left(\mathbf{W}_d\right)_{\ell,k}}. \tag{2.14}$$

where ℓ and k are indices which represent the specific decoder weight in the matrix $\mathbf{W}_d$. In a similar way, we can find the partial derivatives of the objective function with respect to the inputs of the decoder,

$$\frac{\partial \mathcal{L}}{\partial \left(\mathbf{Y}\right)_{\ell,i}} = \sum_{j=1}^{d} \frac{\partial \mathcal{L}}{\partial \left(\hat{\mathbf{X}}\right)_{j,i}} \frac{\partial \left(\hat{\mathbf{X}}\right)_{j,i}}{\partial \left(\mathbf{Y}\right)_{\ell,i}}, \tag{2.15}$$

which can be used to find the partial derivatives with respect to the encoder weights:

$$\frac{\partial \mathcal{L}}{\partial \left(\mathbf{W}_e\right)_{\ell,k}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p} \frac{\partial \mathcal{L}}{\partial \left(\mathbf{Y}\right)_{j,i}} \frac{\partial \left(\mathbf{Y}\right)_{j,i}}{\partial \left(\mathbf{W}_e\right)_{\ell,k}} \tag{2.16}$$

Modern machine learning frameworks such as Pytorch [47], handle the computation of gradients via automatic differentiation and include gradient-based optimization algorithms that adjust the network parameters to minimize the loss, such as Stochastic Gradient Descent, AdaGrad, or Adam [31]. We discuss our specific choice of optimizer in Chapter 5.

2.4 Convolutional Neural Networks

The architecture discussed in the previous section is known as a fully-connected architecture. Signals can be thought of as functions on some domain. In the case of images, the domain is space. When representing images as vectors, where each pixel position corresponds to a dimension, a fully-connected layer will contain weights in a matrix which map all of the input dimensions to a latent space. For larger and larger input sizes, the number of parameters which need to be stored in order to compute this mapping grows as well. Convolutional Neural Networks (CNN) circumvent this issue by restricting the linear mappings, matrix vector multiplication, to convolutions. CNN exploit the fact that images exhibit spatial correlations, and thus the dimensions in a data vector representing an image are coupled. By using a fixed-size kernel that is smaller than the input image size, CNNs restrict their mappings to local operations in space. The kernel works as a sort of window that moves across the input image and computes the cross-correlation of the input image pixels with a learned kernel of weights. The cross-correlation between an image I and a kernel K is given by:

$$s[i,j] = (I * K)[i,j] = \sum_{m}\sum_{n} I[i+m, j+n]K[m,n] \tag{2.17}$$

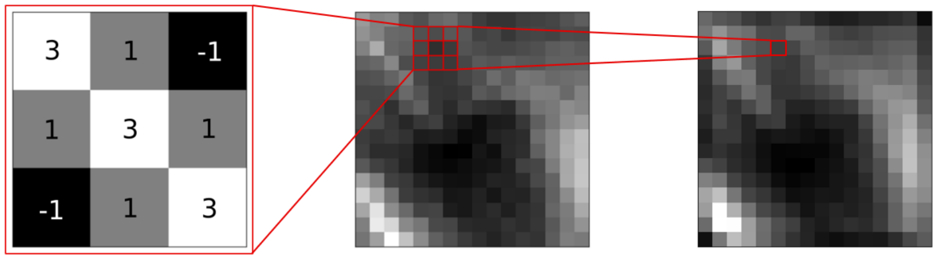


Figure 2.6: A CNN works by learning a kernel, an example of which is shown on the left. This kernel moves across the input image shown in the middle. The kernel elements and local input elements are multiplied element-wise and then summed to produce one pixel of the output on the right. The cross-correlation is computed for all kernel positions, and with a layer of one-pixel padding around the input we can produce an output of the same size as the input. For this example kernel, it tends to emphasize portions of the input where there are edges directed 45 degrees down and to the right.

where i and j represent spatial coordinates on the input image I, and m and n represent the height and width of the kernel [6]. An example of this shifting cross-correlation is given in Figure 2.6.

In essence, we are still computing a weighted sum of input values as we were with fully-connected layers, but we are constraining the weighted sum to only consider the input values around a certain region of the image. The rationale for this is that by moving the kernel over the larger input image, we can still learn valuable features like edges which are present on a small scale in the input. In doing this, we are reducing the number of parameters needed to learn a representation for larger input sizes, as well as reducing the amount of computations. [6]

2.5 Residual Layers

A simple autoencoder like the one described in the previous section may have limited expressive power. Deeper networks can compute more complex functions with complexity growing exponentially with depth [49]. Moreover, our objective is to model an encoding process in multiple stages, namely, retinal processing followed by V1 processing. For each one of these stages, we want our network to be able to approximate sufficiently complex functions that provide efficient encoding of images.

A problem one can run into when training multi-layer models is that of vanishing or exploding gradients. [23] proposed that this issue could be avoided using a learned residual mapping, which is added to an identity mapping of the input to the layer. Given input $\mathbf{x}$, we wish to learn an underlying mapping $\mathcal{H}(\mathbf{x})$. The idea is that we use linear-nonlinear layers in order to approximate complicated functions, so therefore we should also be able to use the same layers to estimate the residual functions given by $\mathcal{H}(\mathbf{x}) - \mathbf{x}$, which is denoted as $\mathcal{F}(\mathbf{x})$.

It has been theorized that many nonlinear layers in sequence can have trouble with approximating $\mathcal{H}(\mathbf{x})$ if it close to an identity mapping. The idea, then, is that

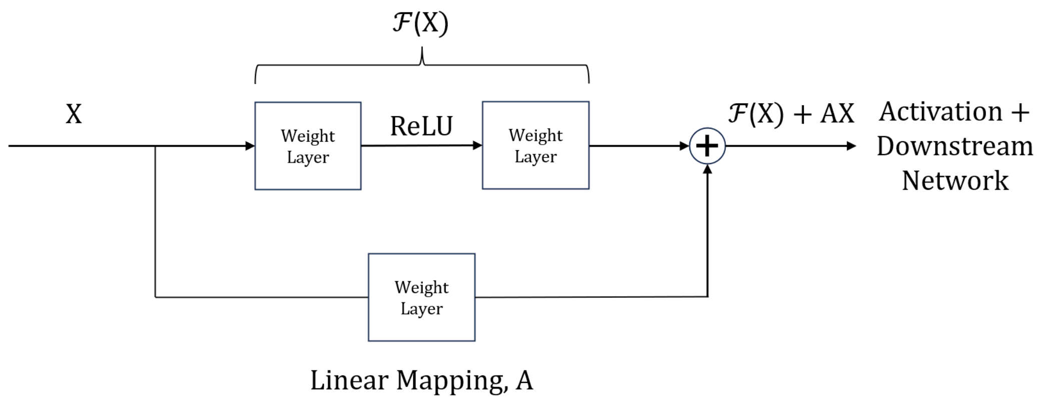


Figure 2.7: A residual layer very similar to the one presented in [23]. Each residual layer contains two weight layers, with a nonlinear activation between them. Rather than an identity mapping, a linear mapping $\mathbf{Ax}$ is added to the residual mapping $\mathcal{F}(\mathbf{x})$. $\mathcal{F}(\mathbf{x}) + \mathbf{Ax}$ is the output of the layer, and therefore represents a mapping of X. This residual module could be used in place of a simple linear-nonlinear mapping to learn a more complex representation.

if the underlying mapping is close to an identity of the input, then the residual $\mathcal{F}(\mathbf{x})$ can be pushed close to zero which allows for a better approximation. While it is improbable that the true underlying mapping is an identity, if the true mapping is closer to an identity mapping than a zero mapping then it helps to precondition the learning [23]. An example of a related residual layer visualization is given in Figure 2.7. In this sense, we should expect that stacking multiple residual layers would only learn residual nonlinearities as needed. The configuration of residual layers allows the flow of the gradient information all the way to the input layer, since the identity mapping which work in parallel with the residual mapping, does not suffer of the vanishing or exploding gradients.

Chapter 3 Related Work

One notable method for training autoencoders on natural images is known as sparse coding [45]. It yields features that resemble a collection of Gabor edge detector receptive fields; these receptive fields exhibit behavior like simple cells in the mammalian V1 [44]. This matches the results of experimentation which determined the receptive field responses of V1 in cats [26] and in macaques [28]. In terms of the retina, the aforementioned center surround receptive fields have been observed in many vertebrates including frogs, rats, lizards, and monkeys [62]. The findings of these physiological studies with primates are often extrapolated to the human visual system, as the results can be correlated to human perceptual studies [37]. The goal of sparse coding is to find a vector $\mathbf{y} \in \mathbb{R}^p$ such that for a set of dictionary elements $\{\phi_i\}_{i=1}^{p}$, where $\phi_i \in \mathbb{R}^d$, the linear combination given by

$$\hat{\mathbf{x}} = \sum_{i=1}^{p} \phi_i \left(\mathbf{y}\right)_i \tag{3.1}$$

is a good approximation of a given input image $\mathbf{x}$. An important feature of $\mathbf{y}$ is that only a few entries of this vector are different from zero. In other words, $\mathbf{y}$ is a sparse vector, which means that information about the input $\mathbf{x}$ can be efficiently represented by $\mathbf{y}$. Given a fixed dictionary $\{\phi_i\}_{i=1}^{p}$ represented as a matrix $\mathbf{\Phi} \in \mathbb{R}^{d \times p}$, the encoding process corresponds to the implicit mapping given by the optimization problem:

$$\mathbf{y} = \underset{\mathbf{a} \in \mathbb{R}^p}{\operatorname{argmin}} \left\|\mathbf{x} - \mathbf{\Phi}\mathbf{a}\right\|^2 + \lambda\Omega(\mathbf{a}), \tag{3.2}$$

where $\Omega(\cdot)$ is the sparsity measure, λ is a tradeoff hyperparameter which determines the importance of the sparsity constraint, and a represents the learned activations which determine which vectors of $\mathbf{\Phi}$ are most relevant. A commonly used candidate for Ω is the L_1 norm,

$$\Omega(\mathbf{a}) = \sum_{i=1}^{p} \left|(\mathbf{a})_i\right|. \tag{3.3}$$

Sparse coding utilizes the principle of redundancy reduction pioneered by Barlow [5] by exploiting high order statistical dependencies beyond linear pairwise correlations. Earlier work in [2] focused on the principle of redundancy reduction in the presence of input noise to model the behavior of retinal ganglion cells. To provide a tractable approach, they use only linear and non-divergent transformations where the the input and output dimensionalities of the model are the same. The model gives rise to receptive fields that resemble those of of RGN cell in primates. Nevertheless, the theory does not explain the choice of the family of transformations and the presence of noise after it.

Doi and Lewicki [18] showed that simply using an efficient coding model was not sufficient to develop an encoding which simulates the receptive field activations of cells in the retina, as the goal of the efficient coding model in the context of the visual system is merely to maximally encode information about the sensory input. They argue that the goal of retinal encoding is more accurately described as trying to encode the sensory signal in such a way that the encoding accounts for degradation of the input signal due to the eye's blurring and noise produced by the photoreceptors themselves. Unlike [2], they also noted that retinal encoding should also seek to be resilient to degradation in the code itself due to neural noise, which occurs during the transmission of the encoding through the optic nerve [18].

Karklin and Simoncelli [30] further developed these ideas by creating a model with accurate noise distributions to describe the noise on the input signal and the output neural noise, as well as utilizing a nonlinear response and metabolic cost on firing spikes. They also use biologically realistic levels of input and neural channel noise when looking at the signal-to-noise ratio.

A popular theory is the idea that via evolution the retina and V1 have both evolved to be as efficient as possible in their encoding method; that is, they preserve the most amount of information as possible from the input via the encoding, subject to any biological constraints such as the metabolic cost of neurons activating [3, 5]. While the efficient coding theory is relatively well-accepted as rationale for vertebrate visual systems developing efficient methods for compressing visual information, there are competing theories on why these methods are different between the retina and V1. Efforts in the past to prove these theories have largely been done either by focusing entirely on the retinal encoding process or focusing entirely on the V1 encoding process. Few papers have focused on developing retinal and V1 encodings in tandem. [53] proposes a model that captures retinal and V1 stages. Each stage uses a composition of sparse PCA followed by ICA and a point-wise nonlinearity aligned with the expected marginal distributions of the ICA outputs. [36] simulates vertebrate retinal and V1 encodings using a model with predominantly convolutional layers in a supervised, task-oriented approach (object classification) rather than unsupervised image reconstruction.

A key finding of [36] is that center surround units can be developed in the retinal layer and Gabor features can be developed in the V1 layer simultaneously by using a dimensional bottleneck at the output of the retinal layer. In the context of autoencoders, a dimensional bottleneck can be instantiated by employing an undercomplete latent space which has a smaller layer width than the input space, then moving to an output or further latent space that is complete or overcomplete. Dimensional bottlenecks in autoencoders can reduce the flow of information, as we have fewer resources to represent the same amount of information. A dimensional bottleneck is a specific instance of the more general information bottleneck. [55] explains based on information theoretic arguments that an information bottleneck can be any constraint that reduces the flow of information. Other forms of reducing flow can be noise in the representation or enforcing high levels of sparsity in the representation.

By enforcing a loss term which maximizes the flow of information from input to output and minimizes the mutual information between the input and the latent space, the bottleneck will learn a representation which only captures the information about the input which is relevant to the task. However, where [55] explicitly reduces mutual information between the input and the latent space, [36] reduces mutual information implicitly via an upper bound imposed by the undercomplete latent space. They ensure the transfer of information between input and output by enforcing a loss term based on classification accuracy. They claim that the primary constraint which led to the development of the appropriate receptive fields in the retina and V1 was the inclusion of the bottleneck, as implementing the bottleneck at the output of the retinal network simultaneously led to the correct receptive fields developing in both the retinal network and the V1 network. The fact that these representations are developed when training on a classification task is interesting, as most work concerned with modeling EVS representations trains using an unsupervised reconstruction task. The reason for using a supervised task may stem from the fact that [36] models further visual processing after the EVS, meaning a classification task could be more apt.

Chapter 4 Method

Our main goal is to construct a model of the early visual system via representation learning. Based on the previous work on sparse coding for V1 and efficient encoding in RGN, we propose a model that is motivated by the informational bottleneck principle. Unlike [36], the information flow in our model is also constrained by input and channel noise, which mimic the noise present before the retina and the noise within the retina and V1 respectively; the choice of nonlinearity; and a metabolic cost term in the objective function which mimics the need for neurons to conserve information by minimizing firing rate. Below we provide more detail on the considerations and choices we made to create the EVSNet.

4.1 Network Architecture

An important consideration when setting up the network to mimic the early visual system is the inclusion of a bottleneck when moving from the retinal network to the V1 network, as found in [36]. As they are using a CNN architecture, the bottleneck ratios they use are 32 retinal channels to 1 retinal channel to 32 V1 channels (32 : 1 : 32), 32 : 2 : 32, and 32 : 4 : 32. In our case we are using fully-connected layers, as we did not want to make the assumption that the learned features will be spatially equivariant. Rather, we use the fully-connected layers in order to let the network produce spatially equivariant features if these are indeed optimal. Thus, we must necessarily use wider layers to account for the features inhabiting different regions of the receptive field.

Initially, we used a network with a fully-connected retinal layer of 100 units and a fully-connected V1 layer of 256 units, so our ratio was 256 input units to 100 retinal units to 256 V1 units. The ratio of V1 neurons to retinal neurons is biologically accurate to the neuron ratio present in some mammals such as cats and macaques [29]. After the retinal and V1 encoding layers, we use a single-layer linear decoder to produce a reconstruction of the inputs; the rationale for choosing a simple linear decoder is explained in Section 4.3. Similar to sparse coding the decoding is linear and the encoding mapping is nonlinear, but unlike sparse coding our encoding map is explicit.

4.1.1 Noise Injection

To the input of the full network we inject additive Gaussian noise which acts to model the noise present in photoreceptors. We also inject additive Gaussian noise at each of the retinal and V1 layers after the linear operation and batch normalization but before the nonlinear activation. This models the neural noise present in the retina and V1. For the input, we inject Gaussian noise of mean 0 and a variable standard deviation, while for each of the layers we inject Gaussian noise of mean 0 and a variable standard deviation which is usually an order of magnitude larger than the input noise. These levels of noise at the input and neural levels are comparable to the biologically accurate levels of noise presented in [30].

4.1.2 Residual Layers

In order to prep the architecture for future work with deeper networks, we opted to turn these single fully-connected layers into modules. For our experiments we implement residual layers (modules), similar in structure to the residual architecture presented in Figure 2.7. Per the figure, each residual layer contains two fully-connected linear layers and a skip connection, which adds a linear mapping of the input to the layer to the output of the second fully-connected layer. When we instantiate a residual layer, we provide the dimensionality of the input and dimensionality of the output. The first linear layer maps from the width of the input to the width of the output, while the second linear layer maps from the width of the output to the width of the output. The output of the first linear layer goes through a ReLU activation function, which is then passed as input to the second linear layer. The output of this layer is added to the skip connection's linear mapping of the input. This sum is then passed through a nonlinear activation function, the choice of which we will discuss next.

4.1.3 Activation Function

In our initial experiments, we found that the sigmoid activation was more likely to produce center surround receptive fields when used in the retinal layer. We also found that a ReLU activation was good at producing oriented Gabor receptive fields. Following these observations, we pursue further ablations on this configuration that will be described in more detail in the experiments Chapter.

A well-known issue that can arise when using a ReLU activation functions is that for any inputs less than 0, the activation value is 0, and the slope in the negative region is also 0. This can lead to the problem of zero gradients, meaning that certain neurons in the layer will not get any gradient signals to update their parameters [38]. A common solution to this issue is to use a *leaky* ReLU activation instead. The leaky ReLU response is given by:

$$y = \begin{cases} mx & x < 0 \\ x & x \geq 0 \end{cases} \tag{4.1}$$

Like the ReLU activation, a leaky ReLU activation has no bounds on activation level (does not saturate like the sigmoid). When the input is greater than 0, the activation function behaves like a linear function with slope 1, and when the input is less than 0, a slope $m << 1$ is employed. The slope m is a user defined parameter. The shape of the activation curve for a leaky ReLU unit with $m = 0.1$ is shown in Figure 4.1

A traditional leaky ReLU can help overcome the zero gradient problem. However, the presence of negative activations became an issue when we added a term to the objective function which attempts to minimize the activation level, as this would

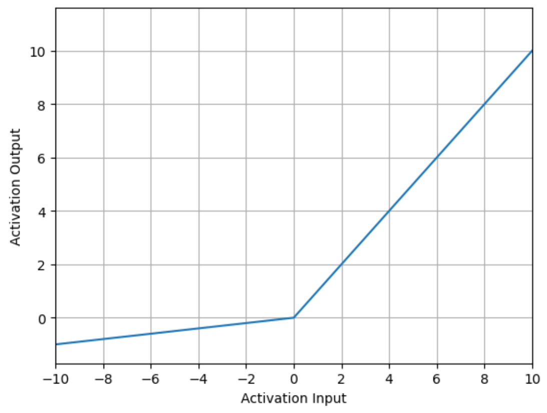


Figure 4.1: The leaky ReLU activation curve, with a slope of 0.1 for the negative x region in order to better visualize the slope in the negative region.

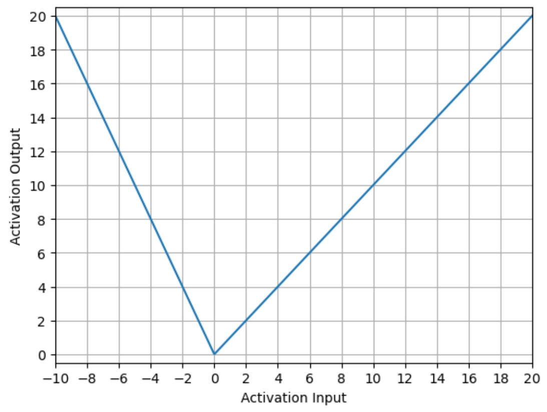


Figure 4.2: Our adjusted leaky ReLU function, which was used when calculating the metabolic cost for each of the retinal and V1 layers.

produce trivial solutions with very large negative activation values. We will explain this more thoroughly in the next section. Attempting to minimize the activation value of the function in Figure 4.1 would mean our weights would keep getting more negative. To combat this issue while still preventing the problem of dead gradients, we instead opt to use a negative slope $m = -2$, as shown in Figure 4.2. This is equivalent to using a regular leaky ReLU with an asymmetric penalty where negative activation would result in a much higher penalty.

Now that we have discussed the model architecture-specific methods we employ, Figure 4.3 shows a block diagram for the proposed model.

4.2 Objective Function

In this section, we describe the main elements that constitute our objective function.

Information Preservation: Usually, the objective function over which the network weights are optimized contains a term which ensures that network maximizes

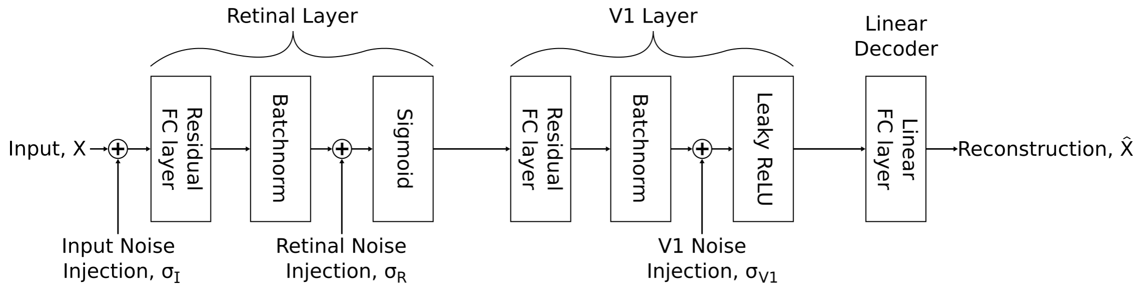


Figure 4.3: In the experimental model, we utilize a residual fully-connected layer in each of the retinal and V1 layers. Notably, we add input noise to the data prior to the retinal layer (σ_I), just before the retinal activation (σ_R), and just before the V1 activation (σ_{V1}).

information transfer between the input and the neuron responses. Ideally such objective would maximize the mutual information $I(X;Y) = H(X) - H(X|Y)$, where X is the random variable representing the input data and Y the neuron responses; however, evaluating the conditional entropy $H(X|Y)$ can be difficult, as the integral used to find it is intractable [61]. Thus, an approximation or surrogate of mutual information is often used [60]. In our case, we take the mean of the aforementioned L_2 norm between the inputs and the reconstructions.

$$L_{\text{rec}} = \mathbb{E}\left[(X_i - \hat{X}_i)^2\right] \tag{4.2}$$

Metabolic Cost: Another common idea motivated by biology is that some portion of the loss function upon which the autoencoder model is trained should include a metabolic cost on firing spikes [36, 30], as an economy of activation energy was suggested in [4]. Namely, the metabolic cost of a neuron is defined as its mean activation multiplied by .

If we are trying to minimize the mean activation level, this requires activation values to be only positive. While this holds for sigmoid and ReLU activations, it is not necessarily satisfied for leaky ReLU activations. As we previously pointed out, the leaky ReLU can help overcome zero gradients that may occur during training with conventional ReLU units, but their use is not compatible with the above definition of metabolic cost. As a solution, we propose an asymmetric function to compute the metabolic cost for leaky ReLU activation layers, shown in Figure 4.2. For the retinal and V1 layers respectively, we do

$$L_{\text{R,meta}} = \mathbb{E}[(Y_{\text{R}})] \tag{4.3}$$

$$L_{\text{V1,meta}} = \mathbb{E}[s_\beta(Y_{\text{V1}})] \tag{4.4}$$

where Y_{R} and Y_{V1} represents the activations of the retinal and V1 layers respectively and,

$$s_\beta(x) = \begin{cases} -\beta x & x < 0 \\ x & x \geq 0 \end{cases} \tag{4.5}$$

with $\beta >> 1$.

Total Cost: In order to build our objective function, we use one reconstruction term, a weight regularization term (L2 regularization as mentioned in 2.12), and metabolic cost terms each for the retinal layer and the V1 layer. It should be noted that the trade-offs for each layer's metabolic cost can be adjusted independently, and their value is represented by λ_{R} and λ_{V1}. Thus, the final objective function takes the following form:

$$L_{\text{total}} = L_{\text{rec}} + \lambda_{\mathrm{R}} L_{\mathrm{R,meta}} + \lambda_{\mathrm{V1}} L_{\mathrm{V1,meta}} + 0.0001 L_{L2reg} \tag{4.6}$$

4.3 Feature Visualization

Visualization techniques have been commonplace in neuroscience to understand how neurons in the visual pathway respond to stimuli. In early stages such as the retina and V1, one popular technique known as spike triggered average has been used to visualize the receptive fields of simple cells [16]. To summarize, the firing of a neuron is recorded when presented with images of i.i.d. samples of white Gaussian noise, and the noise images for which the neuron fires are averaged. This technique can be used to visualize neurons that can be well modeled using a linear-nonlinear model like the following:

$$y(\mathbf{x}) = g(\mathbf{w}^T \mathbf{x}), \tag{4.7}$$

where $\mathbf{w}, \mathbf{x} \in \mathbb{R}^d$ and g is a non-decreasing function such as the sigmoid or the ReLU functions we have discussed. If we limit the energy of the stimuli $\sum_{i=1}^{d} (\mathbf{x})_i^2 \leq E$, the input that maximizes the response $y(\mathbf{x})$ is given by:

$$\sqrt{\frac{E}{\mathbf{w}^T \mathbf{w}}} \mathbf{w}, \tag{4.8}$$

The optimal stimuli is proportional to the weights of the linear transformation in the model neuron. Therefore, for the simple autoencoder with a single encoding layer, we can visualize the learned features by looking at the weights of the rows of the linear transformation matrix in the encoder. However, when dealing with more than one encoder layer, we lose the ability to easily visualize what inputs elicit responses of neurons in layers past the first layer. After all, second-layer weights provide us a mapping from a space the size of the first layer to a space the size of the second layer, which does not tell us directly how the second layer relates to the input.

In our model, not only do we have both a retinal and a V1 stages, but each of these are actually residual fully-connected modules (see Figure 2.7). Each residual fully-connected module contains two regular fully-connected layers, meaning that in total we have four sets of fully-connected layer weights which we are optimizing. We could visualize the first of these layers using the simple visualization method described above, but we'll need to use another technique to visualize the other layers.

There have been various proposals on how to visualize intermediate layers in deep neural networks. An unrefined method for doing so would be to merely look for which images in the dataset provide the maximum activation for each unit in the deeper layer. While this method would be computationally very quick, it lacks nuance. The input image which maximizes an activation may not be unique to a particular unit. When using images from the dataset, there is a lot of correlated information in the image, so it can be tough to discern which part of the input is causing the layer unit to activate highly [41].

Another method involves generating images which maximize the activation of each unit in a layer [19, 39, 43]. Starting from noise in the shape of the input, we can optimize the pixels these noisy inputs with an objective function which tries to maximize a given unit's activation level [19, 39]. A more sophisticated method constrains the inputs to lie in the support of natural images; this is accomplished by optimizing the latent code in a generative adversarial network (GAN) [22] rather than directly optimizing pixel values [43]. However, optimizing pixel values requires adding heuristics like regularization to avoid grainy-looking visualizations. Using GANs has the problem that they usually generate images of entire objects, while retinal and V1 features are known to correspond to features that only cover a small portion of the entire object.

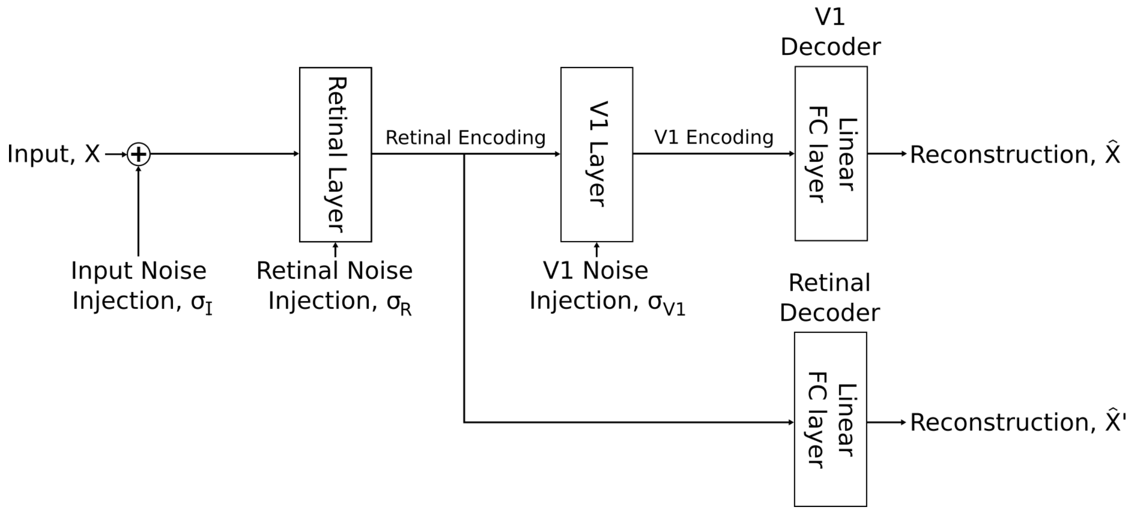
Noticing that our model is inspired by sparse coding (a linear generative model) and that our model contains single-layer linear decoder to reconstruct the input from V1 activations, if we transpose this linear decoder transformation we then have weights providing a linear mapping from the input-space to the V1-space, which is exactly what we want to visualize. In fact, in sparse coding this corresponds to visualizing the dictionary elements [45] which are the column vectors of the matrix $\mathbf{\Phi}$ in (3.2) that span the space of images.

We extend this principle to visualize the retinal features as well. We add another linear decoder to our architecture which decodes the retinal activations back to the input space. The main difference is that this decoder is added to an EVSNet where parameters are kept fixed. Let $\mathbf{y}_{\mathrm{R}} = f_{\mathrm{R}}(\mathbf{x})$ be the output of the retinal module for an image $\mathbf{x}$. Our retina decoder is an affine map,

$$\hat{\mathbf{x}}' = \mathbf{W}_{\mathrm{R}d}\mathbf{y}_{\mathrm{R}} + \mathbf{b}_{\mathrm{R}d}, \tag{4.9}$$

where $\hat{\mathbf{x}}' \in \mathbb{R}^d$ is the input reconstruction from the retina code $\mathbf{y}_{\mathrm{R}} \in \mathbb{R}^{p_{\mathrm{R}}}$, $\mathbf{W}_{\mathrm{R}d} \in \mathbb{R}^{d \times p_{\mathrm{R}}}$, and $\mathbf{b}_{\mathrm{R}d} \in \mathbb{R}^d$. As you can see, similar to the visualization of the decoder weights of the EVSNet, we transpose the weights of this decoder in order to get a mapping from the input-space to the retina-space. Figure 4.3 shows a visualization of how these linear decoders fit into our architecture. The V1 decoder was already in place as the the decoder of our autoencoder, so we add another linear decoder after the end of the retinal layer, which has a different dimensionality due to the different size of the retinal latent space and the V1 latent space.

The retinal decoder can be trained relatively easily, as we employ an objective function which minimizes the MSE loss (2.8) between the input X and the retinal decoder's reconstruction $\hat{X}'$. We also use an L_2 weight regularization (2.12) to make sure the decoder weights are optimized to a unique solution. The training of this decoder is quick, only requiring 100 epochs of training (compared to the thousand



V1 Decoder
Input, X
Retinal Layer
Retinal Encoding
V1 Layer
V1 Encoding
Linear FC layer
Reconstruction, $\hat{X}$
Input Noise Injection, σ_I
Retinal Noise Injection, σ_R
V1 Noise Injection, σ_{V1}
Retinal Decoder
Linear FC layer
Reconstruction, $\hat{X}'$

Figure 4.4: The retinal decoder does not affect any of the network weights or biases during training of the EVSNet. This retina decoder is only trained on the intermediate outputs of the EVSNet while keeping the parameters of the EVSNet model fixed.

epochs required for training the full network). The retinal decoder training does not significantly increase the time required to train the full network, so we can train this decoder several times throughout the EVSNet training to get a visualization of how the retinal features evolve throughout the training process.

4.4 Dataset Generation

When attempting to model the vertebrate visual system, a common dataset is the van Hateren Natural Image Dataset [57], which contains 4,167 images of natural landscapes. The images in this dataset have resolution of (1,024 × 1,536 pixels). Because our focus is on early visual system where receptive fields are known to be localized and occupy a relatively small portion of the visual field, to generate our dataset we take a sample of patches from each image in the dataset. These patches are much smaller in size. Each patch is 16 × 16 pixels, and we pull a user-specified number of patches from each image in the dataset. We also placed a constraint on which patches were picked; patches needed to have a pixel standard deviation larger than a threshold value to ensure a high enough contrast, allowing the autoencoder to better learn localized features.

For each image file in the van Hateren Dataset directory, its data is copied to an array and reshaped to $1{,}024 \times 1{,}536$. Then we run a while loop that iterates until the number of patches taken from this image reaches the user-chosen number of patches per image. During each run of this while loop, a random column and row value is chosen. The column value is chosen between 0 and 1,520 ($1{,}536 - 16$), while the row value is chosen between 0 and 1,008 ($1{,}024 - 16$). The patch is then indexed from the row value to the row value plus 16, and from column value to column value plus 16. The patch values are divided by the patch's maximum value to scale the values to the range [0,1], and then the standard deviation of the patch is checked. If the standard

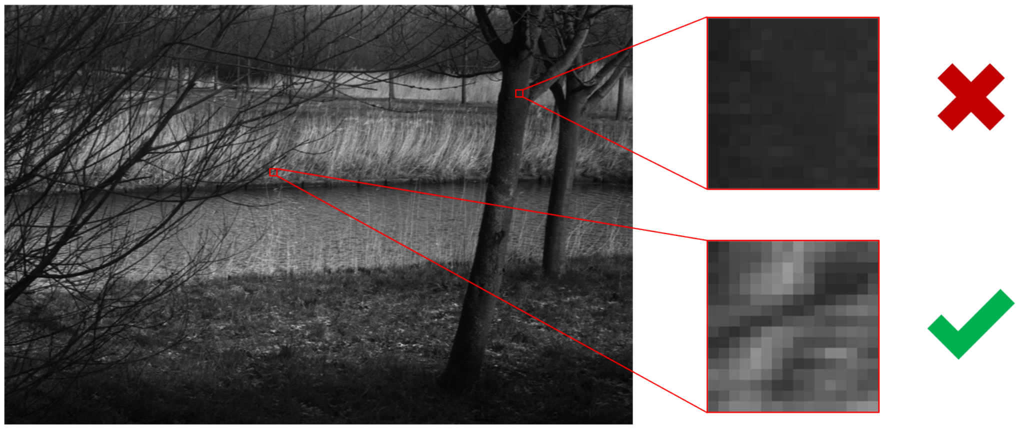


Figure 4.5: At left is one of the full 1024×1536 images from the van Hateren dataset. The two example patches at right represent patches that could be randomly selected for consideration in the patch dataset. For a contrast threshold of 0.1, the patch at top right would be rejected due to a lack of contrast (standard deviation = 0.018), while the patch at bottom right would be accepted as it has a high enough contrast (standard deviation = 0.11).

deviation is lower than a threshold value (i.e. the patch lacks adequate contrast), then the patch is rejected and the loop will run again. If the patch is accepted, the count of accepted patches increases and the loop condition is evaluated.

If the patch has an adequately high standard deviation, then the patch is saved to the array of current patches and the patch counter is incremented by one. The loop will continue to run until the specified number of suitable patches are taken from the current image. These patches are then appended to the array of all patches. The next image is loaded from the directory, and the process repeats until all images have had patches collected from them. An example illustrating the acceptance/rejection of patches is shown in Figure 4.5.

To generate the patch dataset used for the results of this paper, we select 20 patches per image for a total of roughly 83,000 patches, and a contrast threshold value of 0.1. The threshold value was chosen by generating data with a selection of threshold values, and choosing the highest threshold value which caused only negligible change in the patch generation run time when compared to generating without a threshold. The generated training patch dataset was only generated once and it is used for all the experimentation training present in this thesis.

We also generated a separate validation dataset, which took 30 random patches per image for approximately 125,000 patches. The validation dataset was used to discern how many epochs to train the models in our experimentation. Since we are learning with an unsupervised task an evaluating our representation with a separate metric, we do not use a test dataset.

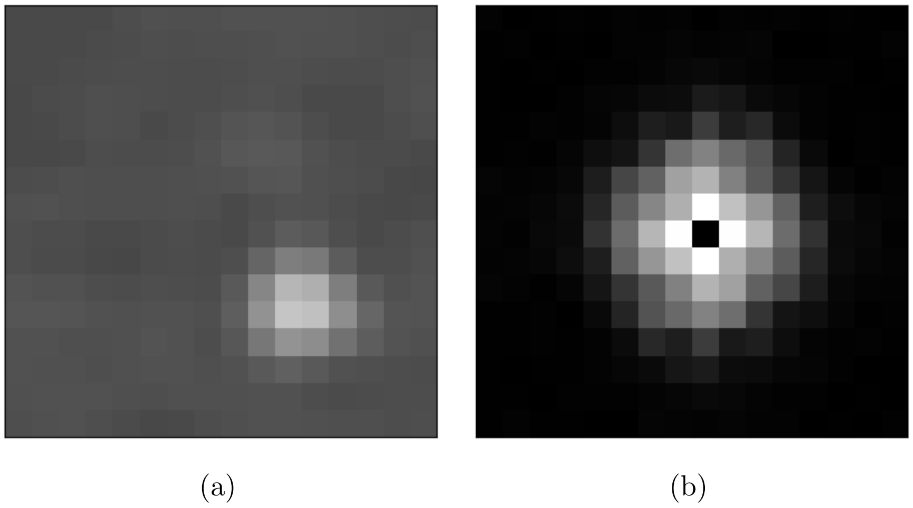


Figure 4.6: A center surround feature and its corresponding 2D FFT. We would describe this center surround feature as unoriented because its 2D FFT is circular.

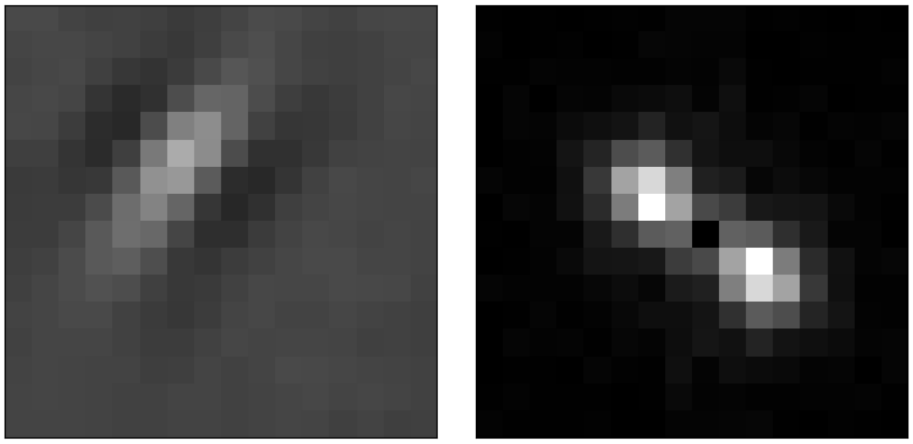
4.5 Orientation Measure

To quantitatively differentiate center surround receptive fields from Gabor receptive fields, we propose our own measure of orientedness. For each of the features in either the retinal or V1 layer vis the linear decoder approach, we can compute the 2-dimensional Fast Fourier tranform (2D FFT), which is a discrete Fourier transform corrsponding to a 2-dimensional array of spatial frequencies. If we take the magnitude of this 2-dimensional FFT, we can visualize the frequencies present in the feature. At the center of the 2D FFT is the DC component or bias of the feature, which is 0 for our cases as the mean value of the features is 0. Moving out from the center of the array, pixel values represent higher frequency components. Pixels near the center of the visualization therefore represent the lower frequency components. As you rotate around the image, you change the direction in which there is a spatial frequency. For example, in Figure 4.6 the FFT pixel values are relatively radially symmetric, as there is no strong orientation in the center surround feature. In Figure 4.7, there are only high values for frequency components along the line $y = -x$, which corresponds to the direction over which the Gabor feature varies.

By interpreting the magnitudes the 2D FFTs of the retinal and V1 features as distributions on the x, y plane, we can construct a heuristic measure of how oblong the distribution is based on 2×2 covariance matrix that is obtained form the coordinate distribution. To construct the covariance matrix A we estimate is entries A_{XX}, A_{XY}, A_{YX} and A_{YY}. Note that, $A_{XY} = A_{YX}$, so we only need to find the covariance term and both variance terms. We find A_{XX} using

$$A_{XX} = \sum_{x=-8}^{7} \sum_{y=-8}^{7} F(x,y)xx \tag{4.10}$$

where $F(x, y)$ represents the value of the magnitude of the 2D FFT at the coor-



(a) (b)

Figure 4.7: We can describe this Gabor as oriented due to the oblong shape of its 2D FFT.

dinates (x, y). Similarly, to find A_{YY} we do

$$A_{YY} = \sum_{x=-8}^{7} \sum_{y=-8}^{7} F(x,y)yy \tag{4.11}$$

Finally, to find A_{XY} we use

$$A_{XY} = \sum_{x=-8}^{7} \sum_{y=-8}^{7} F(x,y)xy. \tag{4.12}$$

We arrange the above terms in a covariance matrix as follows:

$$A = \begin{bmatrix} A_{XX} & A_{XY} \\ A_{YX} & A_{YY} \end{bmatrix} \tag{4.13}$$

which can be used to obtain an ellipse C via the following quadratic equation:

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} A_{XX} & A_{XY} \\ A_{YX} & A_{YY} \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \end{bmatrix} = C. \tag{4.14}$$

The oblongness of the ellipse defined by (4.14) can be determined based on the eigenvalues of A. The larger of the two values represents the variance in the direction of the major axis, while the smaller value represents the variance in the orthogonal direction which corresponds to the minor axis. We can then take the ratio of the larger eigenvalue to the smaller one to get our measure for orientation. For a 2D FFT where the magnitude is rotation invariant, we expect that the primary and secondary variances would be very similar, thus the above defined ratio would be close to 1. For a more oblong 2D FFT (like the one in Figure 4.7), we expect that the major axis variance would be larger than the variance of the minor axis, yielding a higher ratio.

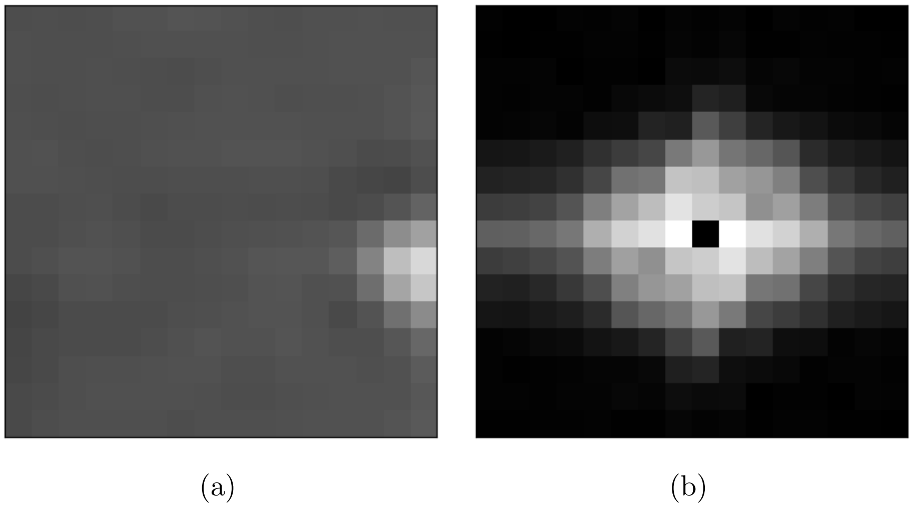


Figure 4.8: A center surround feature on the edge of the receptive field and its corresponding 2D FFT. Despite the fact that it should be unoriented, we see that the 2D FFT appears to be oriented and not circular.

One limitation of the proposed method arises when we look at center surround features that appear near the edges of the patch. We can see in Figure 4.8 that we have what would normally be a center surround feature, but it is cut off on the right edge of the patch area. This causes an artifact in the 2D FFT that introduces extra energy on spatial frequencies along the x-axis. However, we believe this issue can be overlooked as it only affects a few of the receptive fields. Moreover, we can construct a histogram of the orientedness values for all features in each layer, and compare the relative frequency distributions for each layer.

Figure 4.9 shows that we can see a clear difference in the distributions of relative frequencies of orientedness values between the center surround and Gabor features; the center surround features have many more orientedness values close to 1.0, making the distribution much more skewed than the Gabor feature distribution. We compute our orientedness histograms on the range $[1, 7]$ with 20 bins. We can use the chi-squared distance to discern how different the retinal and V1 distributions are, as it is a common method for comparing histogram distributions.

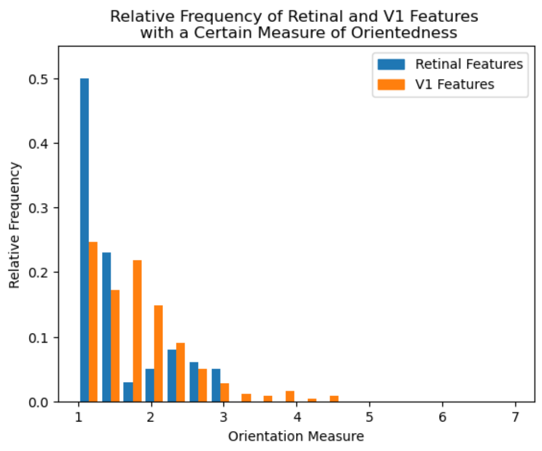


Figure 4.9: The relative frequency histogram for the retinal and V1 features in a model trained with $\sigma_I = 0.1$, $\sigma_R = 4.0$, $\sigma_{V1} = 0.5$, $\lambda_R = 2.0$, and $\lambda_{V1} = 2.0$. The retinal layer has mostly center surround features and the V1 layer has mostly oriented Gabor features. When we are comparing the orientedness of the retinal layer to the V1 layer in this case, we should expect that the relative frequency distributions would differ. This appears to be the case, as the retinal center surround feature distribution appears much more right skew than the V1 Gabor orientedness distribution.

Chapter 5 Experimentation

The proposed model has a set of hyperparameters that control its behavior. Namely, we have the nonlinearity at the retinal and V1 outputs, the input and channel noises (retina and V1) σ_{R} and σ_{V1}, and the metabolic constraint which is indirectly set via tradeoff coefficients $\lambda_{\mathrm{R,\ meta}}$ and $\lambda_{\mathrm{V1,\ meta}}$ in (4.6). Experimentation consisted of training many different EVSNets over a sweep of these hyperparameters, visualizing the learned representations, and quantitatively analyzing these visualizations using the method presented in Section 4.5. We wanted to gain an understanding of how each hyperparameter affected the learned representation.

Note, the hyperparameters above are not the only ones that can be modified, they are the ones that matter the most in changing the behavior of the model. We also varied the sizes of the retina and V1 layers. In order to pick the learning rate, we trained networks over a selection of different learning rates ranging logarithmically from 1e−2 to 1e−6. We then picked the largest learning rate that would still produce stable representations, which was 1e−4. This learning rate was used with an Adam optimizer for all sweeps.

For all sweeps, networks were trained for 1000 epochs each. This training length was chosen based on the loss curves for the networks, for which the change in loss per epoch becomes negligible at this point. As can be seen in Figure 5.1, the training loss curves flattens out considerably after 1000 epochs.

Our method for naming models stems from all the parameters which we adjust. An example of a model name is S_R_100_256_0.1_4.0_0.5_2.0_2.0. The first two letters represent the retinal and V1 activation functions (S for sigmoid and R for leaky ReLU in this case). The next two numbers represent the size of the retinal and V1 layers (100 and 256 units), respectively. The next three numbers represent the values of σ_{I}, σ_{R}, and σ_{V1} respectively (0.1, 4.0, and 0.5). Finally, the last two numbers represent the values of λ_{R} and λ_{V1} (both 2.0).

5.1 Input Noise Sweep

A sweep was conducted over all three noise parameters, with the sweep parameters being σ_{I}:[0.1, 0.5, 1.0], σ_{R}:[0.5, 2, 4], and σ_{V1}:[0.5, 2, 4]; retinal and V1 metabolic tradeoff was set to 1.0. We wanted to observe specifically how each noise parameter affected the learned representation. Specifically, we noticed that the input noise had a very noticeable effect on the learned representation. Shown in Figures 5.2 and 5.3 are the retinal and V1 features for 3 models which had the same parameters, except for the input noise which varies across all 3 possible values.

The main takeaway from the input noise sweep is that increasing the input noise seems only to make the learned features more diffuse, sometimes to the point of losing a discernible localized feature. For this reason, we kept the input noise static at a standard deviation of 0.1. The reason why receptive fields are more diffuse at high level of input noise can be explained from the nature of the frequency distribution of

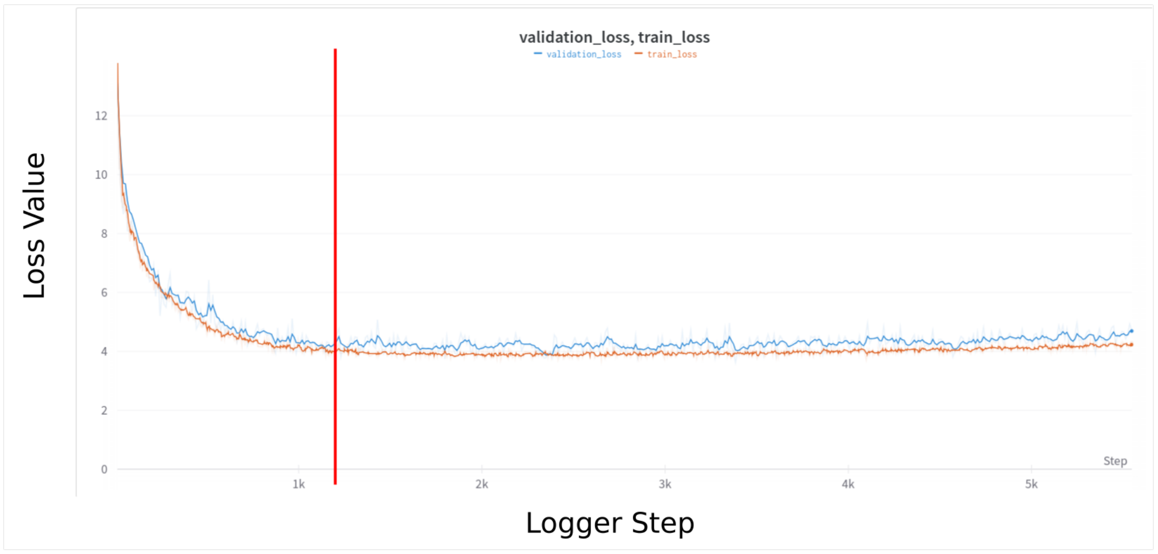


Figure 5.1: An example loss curve, taken from model S_R_100_256_0.1_4.0_0.5_2.0_2.0 which was run at 5000 epochs, with loss on a validation set being computed every 10 epochs. It should be noted that logging each epoch and logging the validation loss count as a logger step, shown on the x-axis. The validation loss is shown in blue, while the train loss is shown in orange. Both curves seems to flatten at around 1500-2000 epochs; however, since we are computing many sweeps over different sets of parameters, the time to compute was very important to us. With this in mind, we opted to use 1000 epochs as our training length, which is shown by the red vertical line.

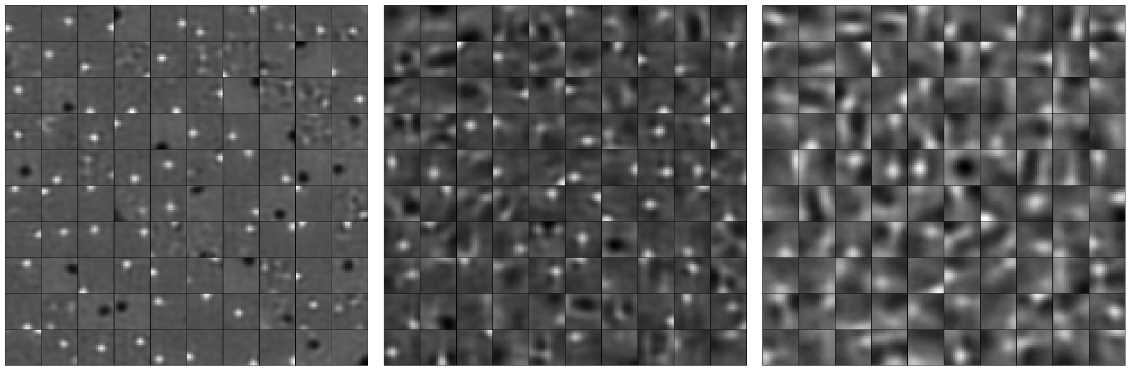


Figure 5.2: As we increase the input noise, the learned retinal representation shows more diffuse features. For 3 models with identical hyperparameters except for input noise (of the form S_R_100_256_XX_2.0_4.0_1.0_1.0), (a) shows the retinal representation with input noise of standard deviation 0.1; (b) shows the representation with with input noise of standard deviation 0.5; and (c) shows the representation learned with input noise of standard deviation 1.0.

natural images which follows a power law $\frac{1}{f^{2-\eta}}$ with η being a small positive number

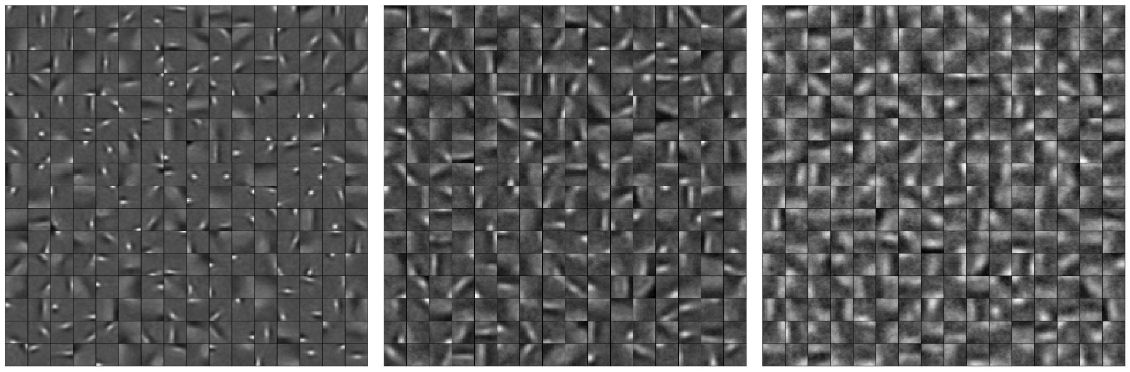


Figure 5.3: The V1 representations for the same 3 models presented in Figure 5.2. Again, as input noise increases from a standard deviation of 0.1 to 0.5 to 1.0, the learned representations become much more diffuse.

Hyperparameter	**Values**
σ_{R}	[0.5, 2, 4]
σ_{V1}	[0.5, 2, 4]
λ_{R}	[0.5, 2]
λ_{V1}	[0.5, 2]

Table 5.1: Default values for hyperparameter sweeps in EVSNet training.

much smaller than 2 [51]. As the noise floor increases, the image information will be only discernible at low frequencies which are above the noise floor.

5.2 Default EVSNet Sweep

After seeing the effects of changing input noise, we opted to keep input noise static at $\sigma_{\mathrm{I}} = 0.1$ for all future sweeps. With this, we set up our default model sweep, which swept over the values in Table 5.1. Retinal and V1 features from a default EVSNet are shown in Figure 5.4. In the retinal layer, we see the appropriate center surround features have developed, while oriented Gabor receptive fields have developed in the V1 layer.

5.3 Coverage Map

Visual assessment of the features in Figure 5.4a show localized features that appear to be translated versions of a few prototypical shapes (center surround receptive fields in our case). This behavior resembles to what we would expect from a translation equivariant representation like the one imposed by using a convolutional layer, where the same filter is applied locally and swept over the entire image [13]. To validate this observation, we make a map which shows the space in the visual field which the center surround receptive fields occupy. To do this, we look at all retinal features and

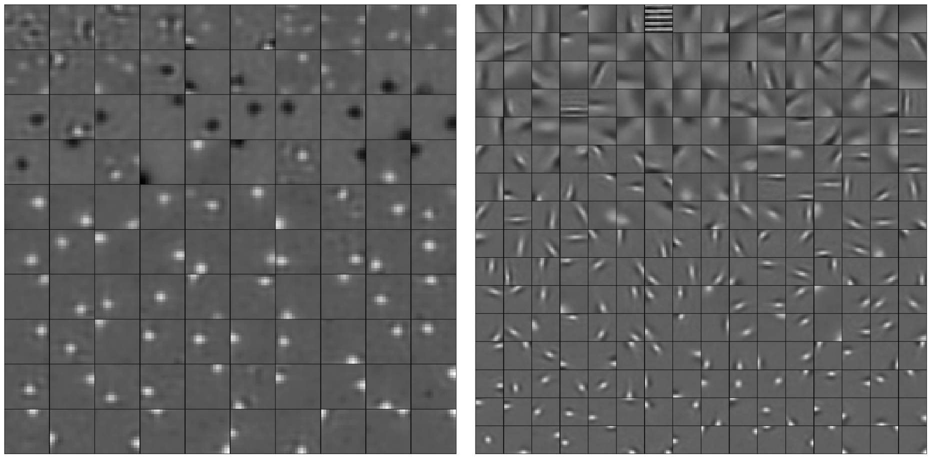

(a) Retinal layer features (b) V1 layer features

Figure 5.4: The learned retinal and V1 representations for model S_R_100_256_0.1_4.0_0.5_2.0_2.0 in the default sweep. We can see clear center surround receptive fields in 5.4a, and oriented Gabors in 5.4b. Both sets of features have been sorted by the absolute value of their skewness (third central moment) as a simple way of ordering the features by diffuseness. This reaffirms that the off-center units are typically more diffuse and fewer than the on-center units. We also see that the V1 features have been roughly ordered by diffuseness as well.

calculate the skewness (third central moment) of their pixel value distributions. We take features with skewness values greater than 1 to correspond to on-center units (for off-center units, we pick all retinal features with a skewness less than -1); doing so removes the retinal features which do not contain prominent localized center surround features. We can take a contour of the center surround units at a threshold value which was picked by matching the contours to their respective unit and adjusting the threshold until the contour matched the size of the feature. We can then draw all the on-center unit and off-center unit contours on two separate fields, and we get two coverage maps like those shown in Figure 5.5. Note the same threshold is applied to all features with high skewness (on-center) and another threshold to all features with high negative skewness (off-center).

The coverage map shows that indeed similar features are evenly distributed across space. We can see that the on-center units, which are smaller and more plentiful than the off-center units, effectively tile the entire visual field with some overlap between units. We also see that the off-center units attempt to spread and cover the visual field as well. This lends credence to the idea that a spatially equivariant representation is optimal when modeling the retina, as we see similarly-shaped receptive fields being repeated and essentially translated across the visual field. It is known that the correlation between the luminance values between two points over natural images only depends on the distance between the two points and not on their specific locations in the image [51]. In other words, the correlation function is translation invariant. That the extracted features are consistent with this characteristic of natural images

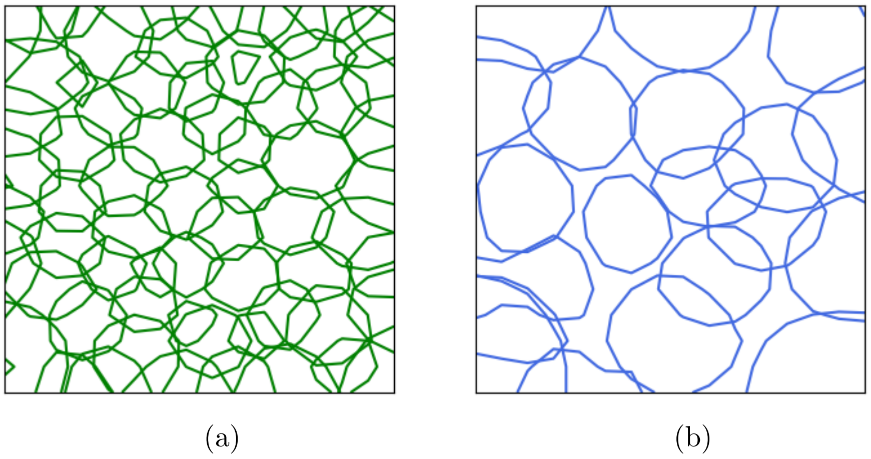


Figure 5.5: The coverage maps for the retinal features in Figure 5.4. (a) shows the contours taken for the on-center units, which tile the entire visual field. (b) shows the contours taken for all the off-center units, which are larger and fewer in number, but still make an attempt to cover the entire visual field. These results support the idea that a spatially equivariant retinal representation is indeed optimal.

can serve as an indication that the proposed model is not overfitting.

5.4 Activation Function Validation

As mentioned before, we observed that sigmoid activations in the network have a tendency to produce center surround receptive fields, while leaky ReLU activations have a tendency to produce oriented Gabor receptive fields. In order to more formally validate this claim, we decided to train two other versions of the network, one of which had sigmoids for both nonlinear activations, and the other of which had our leaky ReLU's for both nonlinear activations.

To see whether the choice of nonlinear activation function is the driving force which pushes the development of center surround or oriented Gabor receptive fields, for both models, we do a sweep over the same set of hyperparameters as the default EVSNet model.

5.4.1 Double Leaky ReLU Model

When looking at features from models in the double leaky ReLU sweep, qualitatively we see that both the retinal and the V1 features appear to be oriented Gabors, as can be seen in Figure 5.6. In order to quantitatively assess that both layers learn the same features, we compute the orientedness measure via the 2D FFTs of the features (as mentioned in 4.5) and then construct a relative frequency histogram of the orientedness values. An example histogram from one of the modes in the sweep can be seen in Figure 5.7. The orientedness relative frequency histogram shows two distributions which appear to be relatively similar, which qualitatively validates

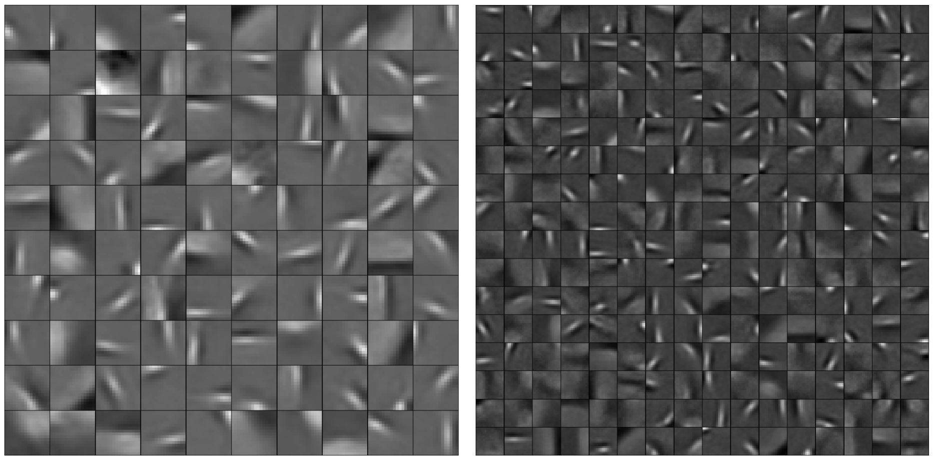

(a) (b)

Figure 5.6: The learned representations for the double leaky ReLU sweep model R_R_100_256_0.1_0.5_4.0_2.0_2.0. The retinal representation when using a leaky ReLU activation at each layer shows clear oriented Gabor receptive fields, despite this not normally being the case. The V1 representation when using a leaky ReLU activation at each layer also shows oriented Gabor receptive fields, which is expected at this layer. Changing the nonlinear activation function for the retinal layer seems to have directly had an effect on what sort of features the layer learns, as this is a single pair of features of many in the sweep which show oriented Gabors in the retinal layer.

that the produced features are similar. Quantitative validation of the distributions' similarity will be explained further in Section 5.6.

5.4.2 Double Sigmoid Model

Similarly, when looking at most of the features in the double sigmoid sweep, we see localized center surround receptive fields in both the retina and V1, as shown in Figure 5.8. We also find the orientedness relative frequency histograms of these features to compare the distributions of orientedness which both layers have. An example histogram from one model in the sweep is shown in Figure 5.9. Similarly to the double leaky ReLU model, the orientedness histogram shows two distributions which are similar, which further qualitatively validates the similarity in learned features. It should be mentioned that the similarity between the retinal and V1 features was observed in all but a few models in the double sigmoid sweep.

For a few specific models in the double sigmoid sweep, we noticed interesting behavior wherein the the V1 features would contain center surround features as well as Gabor receptive fields. This would occur specifically in models which had low V1 noise, and high V1 tradeoff. An example of the edge behavior from the sweep can be seen in Figure 5.10. In order to better show the mix of center surround features and Gabor features, we show the features and the 2D FFTs of the model features, all sorted by their orientedness value in Figure 5.11.

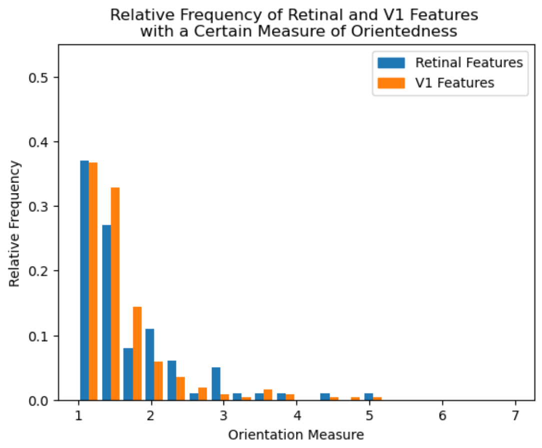


Figure 5.7: The orientedness relative frequency histogram comparing the orientedness of retinal features to V1 features in double leaky ReLU sweep model R_R_100_256_0.1_0.5_4.0_2.0_2.0. Qualitatively, we can see that the two relative frequency distributions look relatively similar, so we can validate our claim that both layers contain predominantly oriented receptive fields.

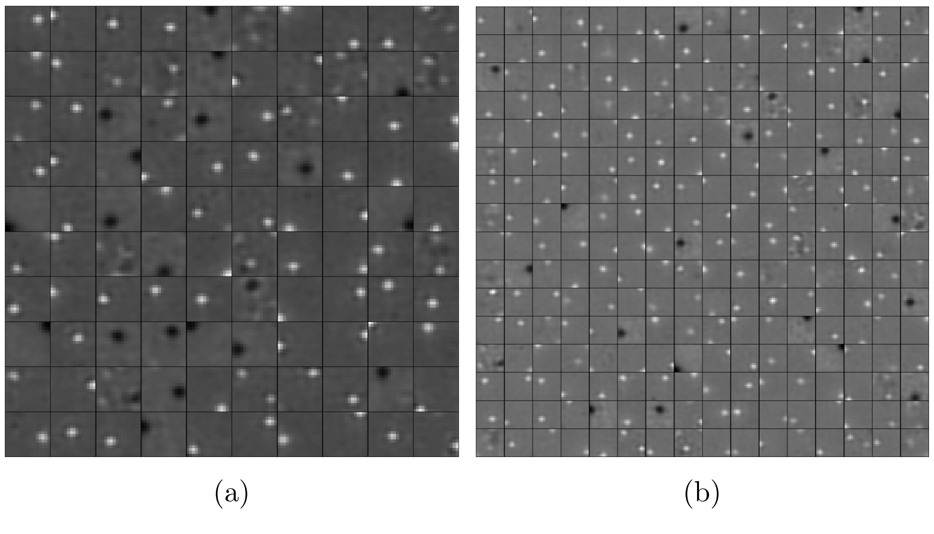


Figure 5.8: The retinal and V1 features for double sigmoid sweep model S_S_100_256_0.1_2.0_4.0_0.5_2.0. Both sets of features appear to be made up of center surround units, which was true of models in the sweep across most parameters, leading us to believe that the choice of nonlinear activation plays a significant role in developing specific receptive fields shapes.

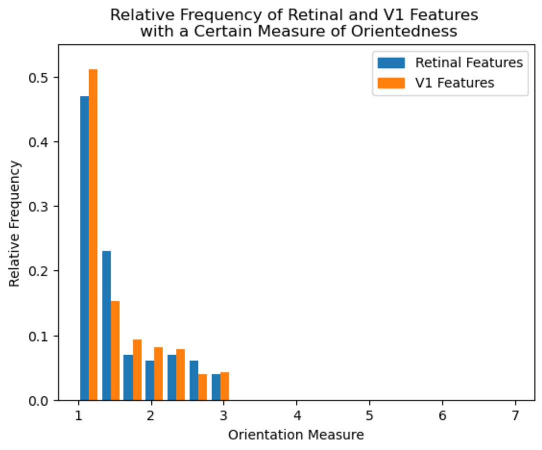
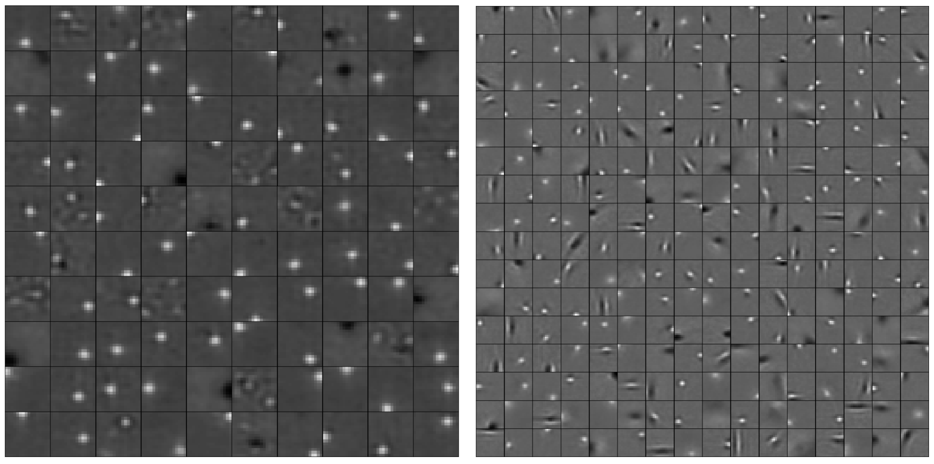
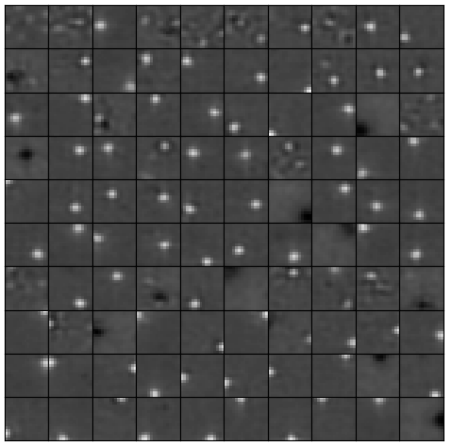


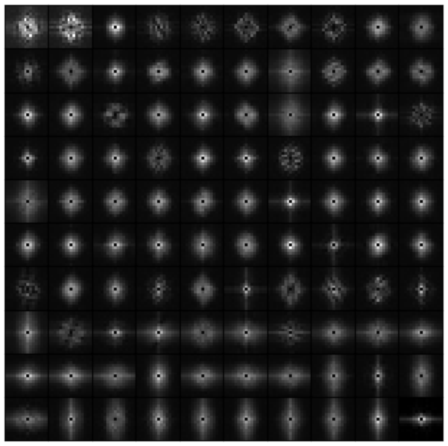
Figure 5.9: The orientedness histogram for double sigmoid sweep model S_S_100_256_0.1_2.0_4.0_0.5_2.0. Both the retinal and V1 features for this model appear to be center surround receptive fields, and the orientedness distributions also seem to agree that both layers have similar features.

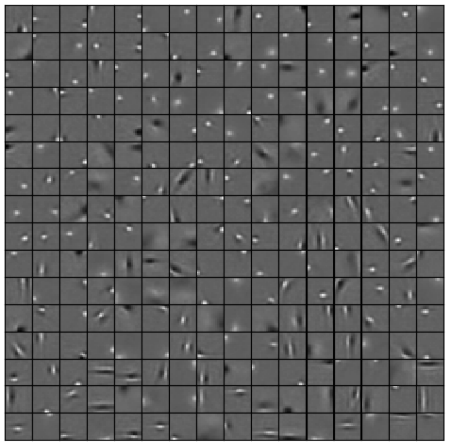


(a) (b)

Figure 5.10: The retinal and V1 features for double sigmoid sweep model S_S_100_256_0.1_2.0_0.5_2.0_2.0. In instances where the sweep pushes the V1 tradeoff high and the V1 noise low, we see that the retinal layer has center surround features as we might expect. However, the V1 layer appears to have a significant amount of oriented Gabor features, in addition to some center surround features. We noted this behavior only occurs when both the V1 noise was low and V1 tradeoff was high.


(a)


(b)


(c)

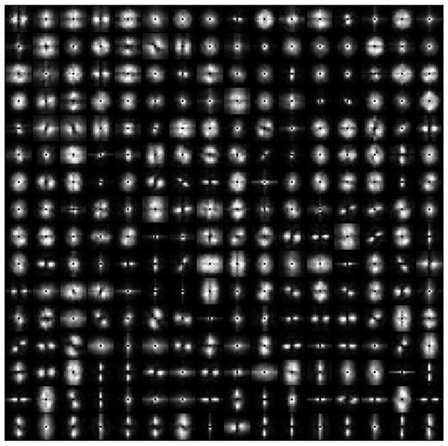
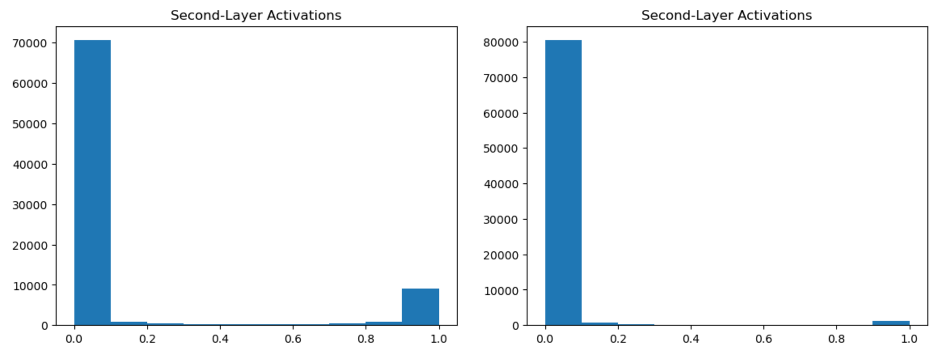

(d)

Figure 5.11: The retinal and V1 features from model S_S_100_256_0.1_2.0_0.5_2.0_2.0 in the double sigmoid sweep and their associated 2D FFTs. Each of these visualization grids has been sorted by the features' orientedness, where the top left element has the lowest orientedness, and you increase orientedness as you move to the right and then down, restarting at the left of each row. We can see in (c) and (d) that there are a few center surround features near the top with relatively low orientation, but as you move down we start to see clearly-oriented Gabor, which is supported by the oblongness of the respective 2D FFTs.



(a) (b)

Figure 5.12: We passed all patches from the dataset through model S_S_100_256_0.1_2.0_0.5_2.0_2.0, and recorded the activations for both a center surround unit and a Gabor unit 5.10b. From this, we constructed a histogram of the two sets of activations; (a) shows the histogram of activations for a center surround receptive field (4th unit), while (b) shows the histogram of activations for an oriented Gabor receptive field (3rd unit). The distributions of activations for center surround receptive fields tend to have more activations in the 1.0 saturation range, as well as more activations distributed in the linear region of the sigmoid; the distribution of activations for oriented Gabor receptive fields tends to be sparser, with many activations in the lowest bin, fewer activations in the 1.0 saturation region, and very few activations in between. While we only show histograms for two specific features, this is a trend which we observed across several sets of center surround and Gabor activation histograms.

To further investigate the edge behavior, we decided to look at the histogram of activations for two specific units in the V1 layer when passing all patches in the dataset through the encoder. Figure 5.12 compares the activation histograms for a center surround feature and a Gabor feature from the V1 features in Figure 5.10. We can see two types of behaviour where one of the histograms shows a higher level of saturation around the upper limit. Since the metabolic cost is the average activation value, the center surround unit has a higher mean activation which would agree with higher metabolic cost and the oriented unit with a lower metabolic cost.

5.5 No-Bottleneck Network

To test the claim in [36] that a dimensional bottleneck at the retinal layer is alone sufficient to cause center surround and oriented Gabor receptive fields to develop simultaneously, we trained a collection of networks with 256 units in both the retinal layer and the V1 layer, removing the dimensional bottleneck that was present in our default EVSNet. We did a sweep across the same set of hyperparameters as the default sweep (see Table 5.1). Features learned from one of the no-bottleneck models can be seen in Figure 5.14. We found that despite removing the dimensional bottleneck, we still learned center surround features in the retinal layer and oriented

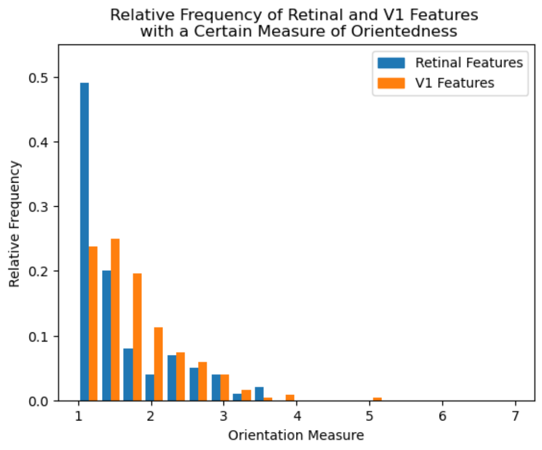


Figure 5.13: The relative frequency orientedness histogram for the double sigmoid model with high V1 tradeoff and low V1 noise S_S_100_256_0.1_2.0_0.5_2.0_2.0 present in Figure 5.10. The two distributions are notably different when looking at the lowest values of orientedness, where there are many more retinal features than V1 features. The difference in distributions makes sense, as more of the V1 features which were expected to be center surround receptive fields have been replaced with oriented Gabor receptive fields.

Gabor features in the V1 layer. The orientedness relative frequency histogram can be seen in Figure 5.15. We can tell the two histogram distributions apart by the difference in high orientedness values; the V1 layer has many more features with high orientedness when compared to the retinal layer's features. This makes sense, as after removing the bottleneck we still developed center surround receptive fields in the retinal layer and oriented Gabor features in the V1 layer. We noticed this distinction in almost all of the models in the no-bottleneck sweep.

For models in the no-bottleneck sweep with high retinal metabolic tradeoff and low retinal noise, we noticed edge behavior similar to what was mentioned in the previous section, where the Gabor features appear at a sigmoid activation (in the retinal layer this time). Again, the Gabor behavior only appears in instances where we have low noise and high tradeoff at a sigmoid layer. The features from one of these models is shown in Figure 5.16. To validate that both layers contain the same oriented Gabor features, the orientedness relative frequency histogram comparison is shown in Figure 5.17. We can see that the distributions of orientedness values look very similar, further validating the similarity in the learned representations.

5.6 Orientedness Histogram Analysis

Over the course of this document we have included several histograms showing the relative frequency of retinal and V1 feature orientedness for specific models. For each of these histograms we qualitatively assess whether the distributions of retinal and V1 orientedness match, but in order to get a quantitative assessment, we take the chi-squared distance between the two distributions for each of these histograms. Table

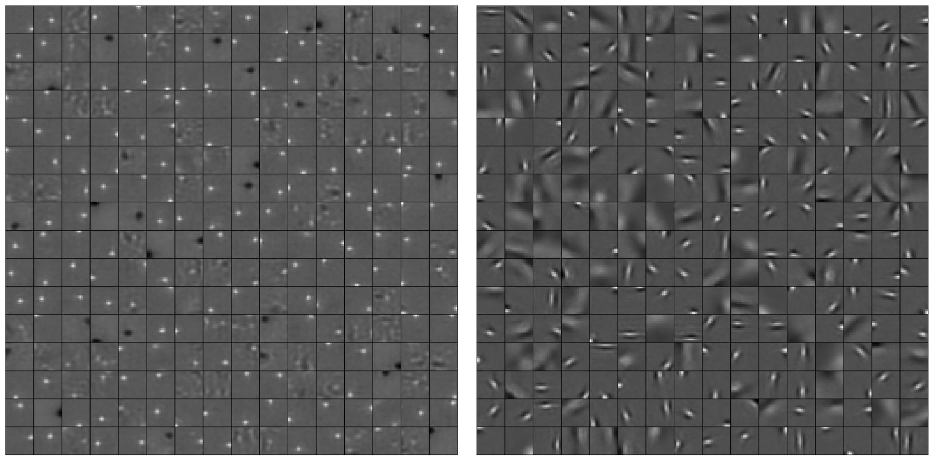

(a) (b)

Figure 5.14: The retinal and V1 features for no-bottleneck sweep model S_R_256_256_0.1_4.0_0.5_2.0_2.0. Despite removing the dimensional bottleneck, we still have learned a retinal representation predominantly composed of center surround features, and a V1 representations predominantly composed of oriented Gabor features. This goes against the idea that a dimensional bottleneck simultaneously encourages both layers to develop their expected features as stated in [36].

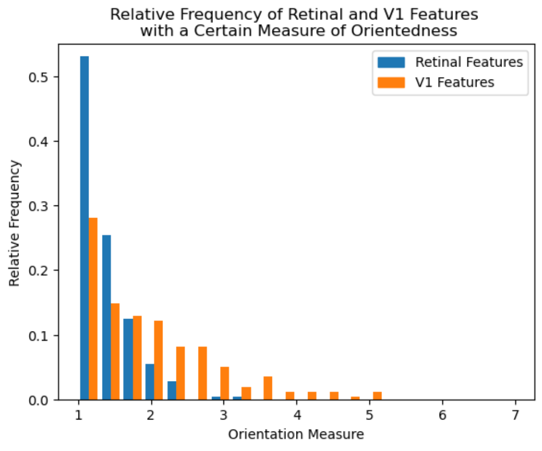


Figure 5.15: The orientedness relative frequency histograms for the no-bottleneck sweep model S_R_256_256_0.1_4.0_0.5_2.0_2.0. When looking at the histograms, we can see that the distributions are very clearly disparate. The low end of the retinal features is much higher than the V1 features, and the V1 features have more high values of orientedness.

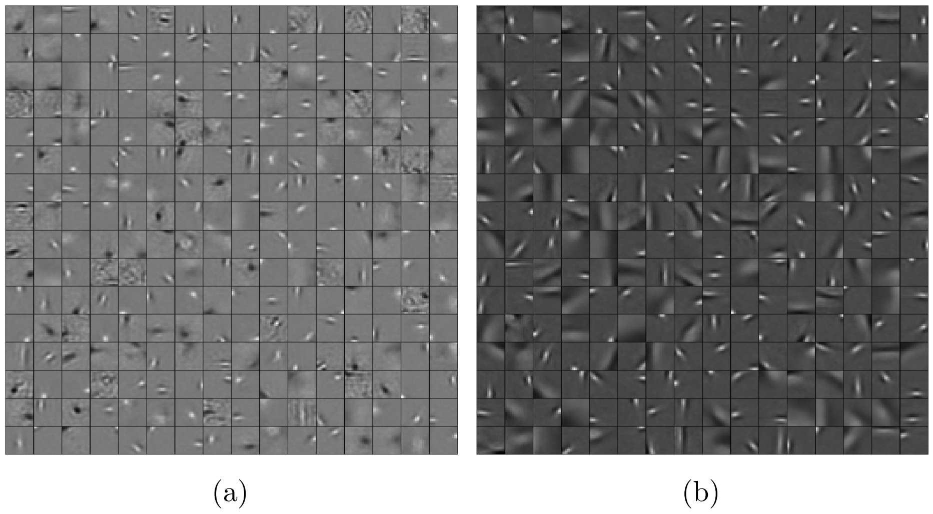


Figure 5.16: The retinal and V1 features for no-bottleneck sweep model S_R_256_256_0.1_0.5_2.0_2.0_2.0. Due to the high retinal tradeoff and low retinal noise, the retinal layer with sigmoid activation learns a representation made predominantly of oriented Gabor features. This is similar to the sigmoid-activated V1 layer learning Gabor features in 5.10.

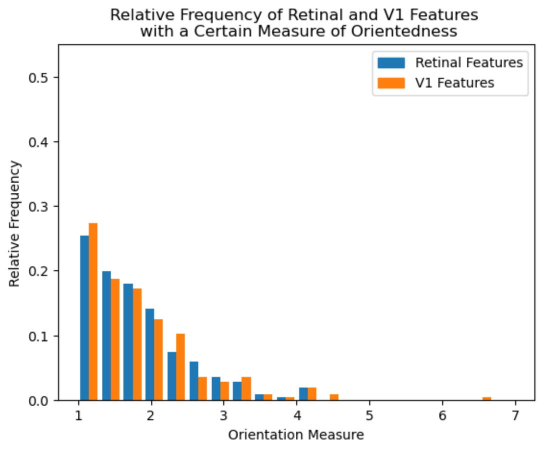


Figure 5.17: The orientedness relative frequency histograms for the no-bottleneck sweep model S_R_256_256_0.1_0.5_2.0_2.0_2.0. To validate that the retinal layer is indeed learning the same features as the V1 layer, we compare the orientedness histograms of both sets of features. Both distributions appear to be the same shape, validating that these features are very similar.

Model Name	Qualitative Assessment	Chi-squared Distance
S_R_100_256_0.1_4.0_0.5_2.0_2.0	Different Distributions	0.342
R_R_100_256_0.1_0.5_4.0_2.0_2.0	Same Distributions	0.0936
S_S_100_256_0.1_2.0_4.0_0.5_2.0	Same Distributions	0.0294
S_S_100_256_0.1_2.0_0.5_2.0_2.0	Different Distributions	0.200
S_R_256_256_0.1_4.0_0.5_2.0_2.0	Different Distributions	0.376
S_R_256_256_0.1_0.5_2.0_2.0_2.0	Same Distributions	0.0260

Table 5.2: A summary of the chi-squared distances of the models and orientedness histograms that were shown specifically in the document. Generally, the chi-squared distance validates the qualitative assessment, as higher values correspond to features or orientedness distributions which appear not to match.

5.2 shows a summary of these assessments, including the model ID, the qualitative assessment, and the chi-squared distance between the two orientedness distributions. The histograms used are Figures 4.9, 5.7, 5.9, 5.13, 5.15, and 5.17.

We notice that for each of the example models where we believe the retinal and V1 features to be coming from the same distribution, the value of chi-squared distance is less than 0.1. For cases where we believe the distributions to be different, the value of chi-squared distance is at or above 0.2.

We must keep in mind that these 6 models only show a few cases where the chi-squared distance works well. Because of this, for the entire default sweep, the double leaky ReLU sweep, and the double sigmoid sweep, we computed each of the 36 models' two orientedness histograms, then found the chi-squared distance between them. The goal was to observe which parameters affected the difference in distributions, and give a more full view of how the chi-squared distance works to differentiate features. We split the graph into 9 groups of models based on the retinal and V1 noise added while training. The color of the bar in the group can be used to determine what the values for tradeoff were during training. The values for chi-squared distance are generally larger in the default sweep when compared to the double sigmoid or double leaky ReLU sweep, which makes sense as we expect the orientedness to be greater when retinal and V1 features are distinct. The results for each sweep can be viewed in Figures 5.18, 5.19, and 5.20.

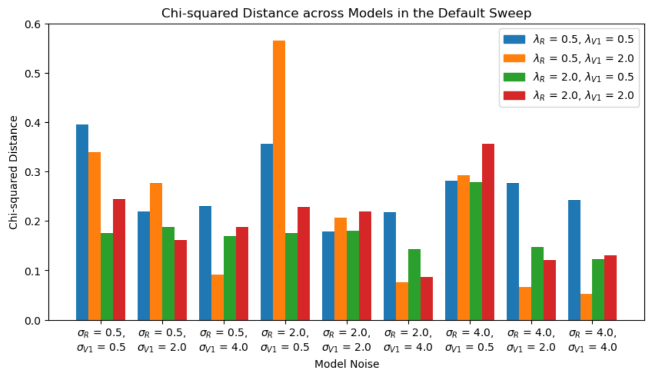


Figure 5.18: This chart shows all the chi-squared distances between the orientedness histograms for each model in the default sweep. We split the graph into 9 groups of models based on the retinal and V1 noise added while training. The color of the bar in the group can be used to determine what the values for tradeoff were during training. These values for chi-squared distance are generally larger than those from the double sigmoid or double leaky ReLU sweep, which makes sense as we expect the orientedness to be different between center surround and Gabor features.

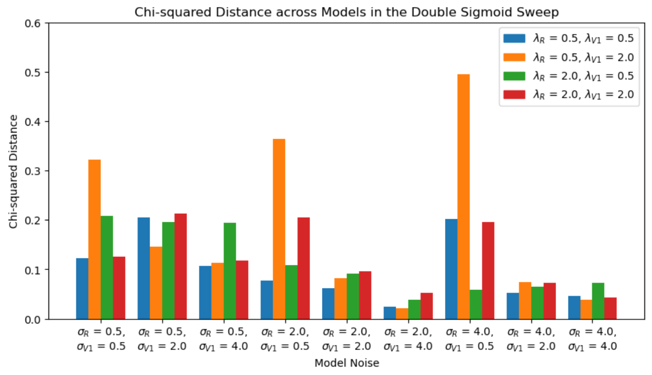


Figure 5.19: The chi-squared distances for the double sigmoid sweep. Generally, these chi-squared distances are relatively low. We do see some significant outliers in the models with low V1 noise and high V1 tradeoff, which follows our observation that such conditions lead to oriented Gabors being learned at a sigmoid layer rather than center surround features.

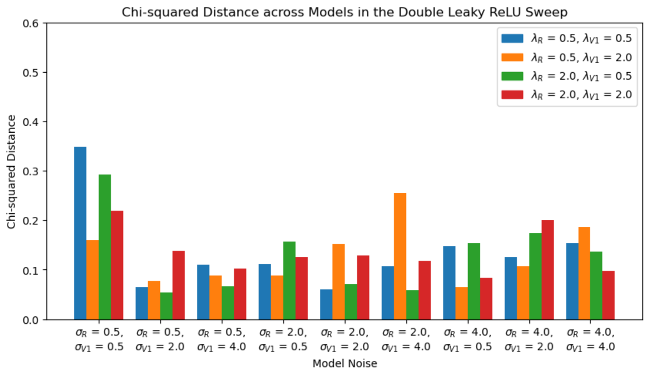


Figure 5.20: The chi-squared distances for the double leaky ReLU sweep. Again, compared to the default sweep, these values for chi-squared distance are mostly lower. This makes sense, as in most cases the retinal layer and V1 layer are learning similar features. There are some instances where there are not as many relevant features learned, which could contribute to some of the higher values of chi-squared distance here.

Chapter 6 Discussion

6.1 Double Activation Models

Per the double leaky ReLU sweep, when we use a leaky ReLU activation at both layers in the EVSNet, the network learns a representation that is predominantly composed of oriented Gabor features at both layers. When looking at the double sigmoid model features in the double sigmoid sweep, we also saw that both the retinal and V1 features are predominantly center surround receptive fields.

We might think, at this point, that the choice of activation function specifically affects the shape of the learned features. However, when we reached certain parameters during the double sigmoid sweep (lowest V1 noise level, highest V1 metabolic tradeoff), we saw a significant amount of the features becoming Gabor receptive fields. Our hypothesis is that this has to do with a lack of high-end activation saturation present when using a sigmoid activation, as is supported by Figure 5.12 which shows that center surround features typically have much higher high-end saturation than the Gabor features when using a sigmoid activation. If high-end saturation is the cause of center surround receptive fields being developed, it would make sense that Gabor edges would develop only in the cases where metabolic cost was high (which pushes the activations to be lower on average) and where V1 noise was low (which pushes fewer activations into the saturation region).

Since we can think of the noise injected between layers as a sort of information bottleneck, we can also think of the presence of an information bottleneck encouraging the development of center surround features; when we limit the noise and use a higher tradeoff which discourages high activation values, we limit the information bottleneck which would normally encourage the learning of center surround features. This observation is consistent with recent electro-physiological studies that support computations that maximize Signal-to-Noise Ratio for information transmission [24].

6.2 No-Bottleneck Model

Through the double sigmoid and double leaky ReLU models, we had found that the shape of the learned representation could be directly affected by the choice of hyperparameters which limit the flow of information. We specifically wanted to test the claim in [36] that a dimensional bottleneck was all that was required to simultaneously develop the appropriate features in both the retinal and V1 layers. We ran the no-bottleneck model sweep, and we found that despite removing the dimensional bottleneck, we still learned center surround features in the retinal layer and oriented Gabor features in the V1 layer. In addition, for a couple specific models with low retinal noise and high retinal tradeoff, we saw that both layers developed oriented Gabor features. As such, we were not able to reproduce the claim presented in [36].

This further cemented to us that it is not specifically a dimensional bottleneck which encourages the development of appropriate features at the retinal and V1 layers, but more generally any sort of constraint which limits the flow of information. While it could be true that the dimensional bottleneck contributed as an information bottleneck in our case (we didn't observe oriented Gabor features in any of the default sweep models, no matter the choice of hyperparameters), we propose that there are many ways one could implement an information bottleneck to develop these features.

6.3 Orientedness

Our proposed measure of orientedness works well when paired with chi-squared distance to quantitatively differentiate center surround features from oriented Gabor features. As seen in Table 5.2, the chi-squared distances between the layer orientedness distributions are significantly higher in instances where qualitatively the features in each layer do not match, and lower in instances where they do match.

When we look at Figures 5.18, 5.19, and 5.20 we can further validate our measure of orientedness. The default sweep has significantly higher values on average, as a majority of the models have a chi-squared distance at or above 0.15, while the double sigmoid and double leaky ReLU sweeps have a majority of models with a chi-squared distance below 0.15. This makes sense, as we expect the default sweep to produce different features in each layer, while for the double activation sweeps we expect similar features in each layer. In the case of the double sigmoid sweep, the models which have a chi-squared distance much greater than 0.2 are the low V1 noise, high V1 tradeoff models which encouraged the development of oriented Gabors in the V1 layer. We didn't observe this sort of behavior in the double leaky ReLU sweep, though there are still a few high values for chi-squared distance. We theorize that this occurred because some of the low noise, low tradeoff models produced fewer localized features in the retina while V1 still produced a good number of oriented Gabors. When computing the orientedness on receptive fields that don't contain localized features (oriented Gabors in this case), we expect there to be less orientedness, meaning that the two distributions of features would be quantitatively farther apart.

6.4 Limitations of the Proposed Model

The proposed model is able to obtain discernible behavior between two layer representing the roles of the retinal ganglion cells and V1. For the retina model, the size of the model provides spatial coverage for the size of input 16×16 pixels, where the center surround behaviour can be observed. However, the proportion of on-center and off-center units does not necessarily reflect the proportion and asymmetries observed in mammalian visual system [11]; we expect that there should be more off-center features learned in the retinal layer.

A limitation regarding V1 is that our proposed model is only able to capture the behaviour of simple cells. A model that include complex cell behaviour would necessitate further layers to produce the invariances observed in complex cells [48, 9].

Contrast normalization is another operation that is known to occur in the early visual system [10]. In the proposed approach the model's ability to include this behavior was neither enforced nor confirmed. It is possible that the composition of multiple layers of computation in the residual module capture some aspects of this kind of nonlinearity, but this was not tested.

Chapter 7 Conclusion

In this thesis, we have proposed the EVSNet model, which learns representations which mimic real-world retinal and V1 representations in the vertebrate early visual system. This model uses fully-connected layers, so as to not assume spatial equivariance of the learned features; additionally we used biologically inspired constraints on learning in order to develop a learned representation which matches the shape of retinal and V1 encodings which have been found experimentally from nature. By utilizing sweeps of hyperparameters and training many unique EVSNet models, we have gained an understanding of how adjusting certain learning constraints can affect the final learned representation. We found that the dynamic range of the responses which can be controlled by the layer nonlinearity, in conjunction with noise and metabolic constraints has a controllable effect on the features that emerge in the model. In order to bolster our qualitative assessment of the difference in expected retinal and V1 feature shapes, we proposed a measure of orientedness, which we use to show differences and similarities between learned retinal and V1 representations. Finally, we compared our findings to the existing work and found that we could generalize the leading paper's findings to say that information bottleneck principles including a dimensional bottleneck could be applied to encourage the learning of biologically accurate retinal and V1 representations.

7.1 Future Work

Future efforts to develop upon the findings of this thesis could be directed towards polishing the orientedness measure. The measure worked well for us to help quantitatively differentiate center surround and Gabor receptive fields when there are not a lot of quantitative analysis methods for this sort of unsupervised representation learning. However, the issue of edge features being represented as more oriented than they should be could be addressed or accounted for.

We found that inhibiting the flow of information with noise and the metabolic tradeoff encouraged the development of the appropriate biological representations. In our case, we used a surrogate for the mutual information in order to judge the transfer of information. It would perhaps be salient to use a mutual information estimator or a bound on the mutual information instead, as we could gain a better idea about exactly how changing the information bottleneck constraints affects how much information is transferred.

We would also like to investigate the role of more sophisticated nonlinearities like divisive normalization, which can also shape the features that characterize both stages of our model [8]. Also, the model was trained purely on reconstruction error. There have been many recent advances in self-supervised learning in computer vision and natural language processing. An open avenue for experimentation is training these models via self-supervised learning which could be better aligned with behavioral tasks that are relevant for survival [46].

Bibliography

[1] J. J. Atick. Could information theory provide an ecological theory of sensory processing? *Network: Computation in Neural Systems*, 22(1-4):4–44, 1992.

[2] J. J. Atick and A. N. Redlich. Towards a Theory of Early Visual Processing. *Neural Computation*, 2(3):308–320, 1990.

[3] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3):183–193, 1954.

[4] H. Barlow. Redundancy reduction revisited. *Network: Computation in Neural Systems*, 12(3):241–253, 2001.

[5] H. B. Barlow. Possible Principles Underlying the Transformations of Sensory Messages. *Sensory Communication*, pages 216–234, 1961.

[6] Y. Bengio, I. Goodfellow, and A. Courville. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

[7] W. Bialek, F. Rieke, R. R. De Ruyter Van Steveninck, and D. Warland. Reading a neural code. *Science*, 252(5014):1854–1857, 1991.

[8] M. F. Burg, S. A. Cadena, G. H. Denfield, E. Y. Walker, A. S. Tolias, M. Bethge, and A. S. Ecker. Learning divisive normalization in primary visual cortex. *PLOS Computational Biology*, 17(6):1–31, 06 2021.

[9] M. Carandini. What simple and complex cells compute. *J Physiol.*, 2006.

[10] M. Carandini and D. Heeger. Normalization as a canonical neural computation. *Nat Rev Neurosci*, 2011.

[11] E. Chichilnisky and R. Kalmar. Functional asymmetries in on and off ganglion cells of primate retina. *Journal of Neuroscience*, 2002.

[12] E. K. Chong, W.-S. Lu, and S. H. Zak. *An Introduction to Optimization, Fifth Edition*. John Wiley and Sons, 2023.

[13] T. Cohen and M. Welling. Group equivariant convolutional networks. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR.

[14] T. M. Cover and J. A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006.

[15] J. Dapello, T. Marques, M. Schrimpf, F. Geiger, D. Cox, and J. J. DiCarlo. Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13073–13087. Curran Associates, Inc., 2020.

[16] P. Dayan and L. F. Abbott. *Tehoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* MIT Press, 2001.

[17] R. L. De Valois, D. G. Albrecht, and L. G. Thorell. Spatial frequency selectivity of cells in macaque visual cortex. *Vision research*, 22(5):545–559, 1982.

[18] E. Doi and M. S. Lewicki. A theory of retinal population coding. *Advances in Neural Information Processing Systems*, (January 2007):353–360, 2007.

[19] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*, 01 2009.

[20] D. J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379, 1987.

[21] D. J. Field. What Is the Goal of Sensory Coding?, 1994.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 2016.

[24] J. Homann and M. A. Freed. A mammalian retinal ganglion cell implements a neuronal computation that maximizes the snr of its postsynaptic currents. *Journal of Neuroscience*, 37(6):1468–1478, 2017.

[25] D. H. Hubel. *Eye, brain, and vision.* Scientific American Library/Scientific American Books, 1995.

[26] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, pages 247–250, 1959.

[27] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160:106–154, 1962.

[28] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[29] J. Jang, M. Song, and S. B. Paik. Retino-Cortical Mapping Ratio Predicts Columnar and Salt-and-Pepper Organization in Mammalian Visual Cortex. *Cell Reports*, 30(10):3270–3279.e3, 2020.

[30] Y. Karklin and E. P. Simoncelli. Efficient coding of natural images with a population of noisy Linear-Nonlinear neurons. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 24:999–1007, dec 2011.

[31] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

[32] N. Kriegeskorte. Deep neural networks: A new framework for modeling biological vision and brain information processing. *Annual Review of Vision Science*, 1(1):417–446, 2015. PMID: 28532370.

[33] S. W. Kuffler. Discharge patterns and functional organization of mammalian retina. *Journal of neurophysiology*, 16(1):37–68, 1953.

[34] S. Laughlin. A simple coding procedure enhances a neuron's information capacity. *Zeitschrift fur Naturforschung - Section C Journal of Biosciences*, 36(9-10):910–912, 1981.

[35] J. Y. Lettvin, H. R. Maturana, W. S. Mcculloch, and W. H. Pitts. What the frog's eye tells the frog's brain. proceedings of the. *Proceedings of the IRE*, 47:1940–1959, 1959.

[36] J. Lindsey, S. A. Ocko, S. Ganguli, and S. Deny. A Unified Theory of Early Visual Representations from Retina to Cortex through Anatomically Constrained Deep CNNs. jan 2019.

[37] M. Livingstone and D. Hubel. Segregation of Depth: Form, Anatomy, Color, Physiology, and Movement, and Perception. *Science*, 240(4853):740–749, 1988.

[38] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.

[39] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196, 2015.

[40] P. Miller. *An Introductory Course in Computational Neuroscience.* Computational Neuroscience Series. MIT Press, 2018.

[41] C. Molnar. *Interpretable machine learning.* Christoph Molnar, 2023.

[42] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[43] A. Nguyen, A. Dosovitskiy, T. Yosinski, Jason band Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *NIPS 29*, 2016.

[44] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images, 1996.

[45] B. A. Olshausen and D. J. Field. Natural image statistics and efficient coding. *Network: computation in neural systems*, 7(2):333–339, 1996.

[46] A. E. Orhan, V. V. Gupta, and B. M. Lake. Self-supervised learning through the eyes of a child. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[48] D. A. Pollen and S. F. Ronner. Spatial computation performed by simple and complex cells in the visual cortex of the cat. *Vision Research*, 22(1):101–118, 1982.

[49] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks, 2017.

[50] M. Rehn and F. T. Sommer. A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of Computational Neuroscience*, 22(2):135–146, 2007.

[51] D. L. Ruderman. The statistics of natural images. *Network: Computation in Neural Systems*, 5(4):517–548, 1994.

[52] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[53] H. Shan and G. Cottrell. Efficient visual coding: From retina to v2, 2014.

[54] M. V. Srinivasan, S. B. Laughlin, and A. Dubs. Predictive coding: A fresh view of inhibition in the retina. *Proceedings of the Royal Society of London - Biological Sciences*, 216(1205):427–459, 1982.

[55] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. pages 1–16, 2000.

[56] M. H. Turner, L. G. Sanchez Giraldo, O. Schwartz, and F. Rieke. Stimulus- and goal-oriented frameworks for understanding natural vision. *Nature Neuroscience*, 25:15–24, 2019.

[57] J. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc.R.Soc.Lond*, 1998.

[58] J. H. van Hateren. A theory of maximizing sensory information. *Biological cybernetics*, 68(1):23–29, 1992.

[59] B. T. Vincent, R. J. Baddeley, T. Troscianko, and I. D. Gilchrist. Is the early visual system optimised to be energy efficient? *Network: Computation in Neural Systems*, 16(2-3):175–190, 2005.

[60] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol. Stacked denoising autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[61] P. Viola and W. M. Wells III. Alignment by maximization of mutual information. *International journal of computer vision*, 24(2):137–154, 1997.

[62] T. N. Wiesel and D. H. Hubel. Spatial and chromatic interactions in the lateral geniculate body of the rhesus monkey. *Journal of neurophysiology*, 29(6):1115–1156, 1966.

Vita

Nicholas Patrick Lanning

Education
Bachelor of Science in Electrical Engineering from the University of Kentucky, August 2017 - May 2021

Employment
Graduate Research and Teaching Assistant, August 2021 - May 2023 Undergraduate Research Assistant, August 2020 - May 2021