Short Papers

Signal Processing in the Retina: Interpretable Graph Classifier to Predict Ganglion Cell Responses

YASAMAN PARHIZKAR, GENE CHEUNG (Fellow, IEEE), AND ANDREW W. ECKFORD (Senior Member, IEEE)

Department of Electrical Engineering and Computer Science, York University, Toronto, ON M3J 1P3, Canada

CORRESPONDING AUTHOR: YASAMAN PARHIZKAR (e-mail: yasaman.parhizkar@gmail.com)

ABSTRACT It is a popular hypothesis in neuroscience that ganglion cells in the retina are activated by selectively detecting visual features in an observed scene. While ganglion cell firings can be predicted via data-trained deep neural nets, the networks remain indecipherable, thus providing little understanding of the cells' underlying operations. To extract knowledge from the cell firings, in this paper we learn an interpretable graph-based classifier from data to predict the firings of ganglion cells in response to visual stimuli. Specifically, we learn a positive semi-definite (PSD) metric matrix $\mathbf{M} \succeq 0$ that defines Mahalanobis distances between graph nodes (visual events) endowed with pre-computed feature vectors; the computed inter-node distances lead to edge weights and a combinatorial graph that is amenable to binary classification. Mathematically, we define the objective of metric matrix $\mathbf{M}$ optimization using a graph adaptation of large margin nearest neighbor (LMNN), which is rewritten as a semi-definite programming (SDP) problem. We solve it efficiently via a fast approximation called Gershgorin disc perfect alignment (GDPA) linearization. The learned metric matrix $\mathbf{M}$ provides interpretability: important features are identified along $\mathbf{M}$'s diagonal, and their mutual relationships are inferred from off-diagonal terms. Our fast metric learning framework can be applied to other biological systems with pre-chosen features that require interpretation.

INDEX TERMS Graph signal processing, interpretability, metric learning, retinal ganglion cell encoding, semi-definite programing.

I. INTRODUCTION

The retina is a thin layer of nerve tissue at the back of the vertebrate eye, which receives images and transmits them as electric pulses through the optic nerve to the brain. It consists of several layers of cells: from photoreceptors which detect light and primary colors, to ganglion cells with long axons stemming from the optic nerve [1]. As such, the retina is an important example of biochemical and neurological signalling, both detecting and preprocessing biological signals before transmitting them to the brain's visual cortex.

In the analysis of retinal signals, one hypothesis conjectures that each ganglion cell type computes one or more specific features of the visual scene through a dedicated neural circuit, which connects it to the photoreceptors [2]. This implies that *the brain's downstream regions do not receive a general representation of the image, but instead receive a highly processed set of extracted features*. For instance, some types of ganglion cells were found responding to features such as motion in a specific direction, texture motion, and anticipation of a periodic stimulus [2]. Thus, it is natural to consider whether those features can be extracted from the firing patterns.

State-of-the-art cell firing prediction algorithms employ deep learning, such as Convolutional Neural Networks (CNN) [3], [4], [5] and Recurrent Neural Networks (RNN) [6]. While deep learning models predict well, they are inherently uninterpretable "black boxes" and fail to reveal biological information about what exactly trigger cell firings. *In response, the main contribution of this paper is to extract knowledge from the cell firings by learning an interpretable graph-based binary classifier from data*. Specifically, we learn a *positive semi-definite* (PSD) metric matrix $\mathbf{M} \succeq 0$ that defines Mahalanobis distances between graph nodes (visual events) endowed with pre-computed feature vectors. The computed node-to-node distances lead to edge weights and a finite graph that is amenable to clustering and binary classification.

Mathematically, unlike the previous *graph Laplacian regularizer* (GLR) [7] used in metric learning [8], we adapt the *large margin nearest neighbor* (LMNN) method [9] into our graph setting as an objective function, and formulate a *semi-definite programming* (SDP) problem [10] to optimize $\mathbf{M}$. We efficiently solve it via an adoption of a fast approximation called *Gershgorin disc perfect alignment* (GDPA) linearization [11]. The learned $\mathbf{M}$ provides a level of interpretability: features crucial to classification can be identified immediately along $\mathbf{M}$'s diagonal, while their mutual relationships can be directly inferred from off-diagonal terms.

The employed features in this paper are localized spatially, and one set (3D-SIFT) has clear interpretation in terms of local image gradients. This can enable the estimation of a ganglion cell's receptive field [12] and type [2] based on the features to which it is most sensitive. This estimation is inferred directly from the natural stimulus data without the need for random noise stimulus data [13] or more complex experiments. More generally, our fast metric learning framework is applicable to other biological systems with pre-chosen features that require interpretation.

We organize the paper as follows. In Section II, we review previous attempts at understanding the retina's visual code. We discuss the shortcomings of these works and establish the need for interpretability. In Section III, we describe our proposed model along with the training and validation processes. We delineate how the Mahalanobis distance metric can be trained by minimizing the GLR, then we discuss GLR's limitations. Section IV manifests the main contribution of this paper: adapting LMNN to the graph setting as a new training objective called *graph-based large margin nearest neighbour* (GLMNN) and approximating the solution using GDPA linearization for faster computation. Experimental results and conclusion are presented in Sections V and VI, respectively.

II. THE VISUAL CODE OF THE RETINA

The retina is a complex network of diverse cells intricately connected to each other in smaller local circuits. These circuits may have various roles, from basic enhancements to the eye's visual input to extracting sophisticated visual features [2]. Many works investigate the cellular circuits of the retina and their role in extracting specific features from the eye's visual input [14], [15], [16], [17]. Although these investigations provide thorough insight into the retina's structure and function, their development is slow and difficult. Another standard approach is the Linear-nonlinear (LN) model [18] which combines a linear spatiotemporal filter with a single static nonlinearity. These models have been used to describe the retinal responses to artificial stimuli; however, they fail to generalize to natural stimuli [19].

Deep learning models significantly outperform traditional approaches in predicting retinal responses both to artificial and natural stimuli as demonstrated in [3], [4], [5], [6]. However, as discussed in [5], the opaqueness of these models makes it difficult to gain insight into the retinal cells' collective or individual behavior. A systematic approach to shed light on the trained deep learning models is proposed in [5] using dimensionality reduction and modern attribution methods.

The approach in [5] falls under a category of interpretability techniques called "post-hoc" methods [20]. These methods attempt to interpret a black-box model after it is designed and trained. Another category is the creation of intrinsically more transparent AI models, including our approach: we employ interpretable components to
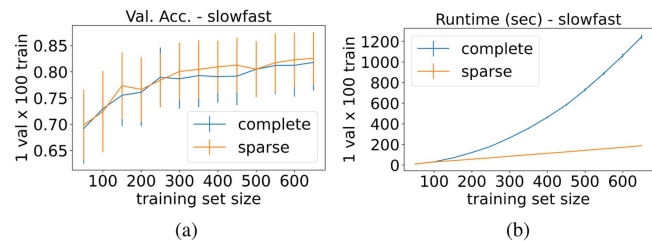

FIGURE 1. Comparison between graph-based classifiers for a *complete* graph and a *sparse* graph using the GLR objective, as function of training dataset size, in (a) prediction accuracy and (b) runtime.

build a more comprehensible classification system, while maintaining competitive prediction accuracy with state-of-the-art methods.

To draw a parallel between our approach and [5], we can compare the attribution scores presented in [5] to the diagonal entries of our computed metric matrix $\mathbf{M}^*$ in our method, which can be further used for feature selection. However, there is no equivalence to the non-diagonal entries of $\mathbf{M}^*$ in [5]—the relationship between features themselves are left unexplored. Moreover, our model provides an intuitive map of all datapoints; the distance of a newly introduced datapoint to known datapoints can be easily inferred and understood.

III. GRAPH METRIC LEARNING AND GLR OPTIMIZATION

We first discuss metric learning on graphs and juxtapose our model with previously developed approaches. Then, we describe the dataset used in our experiment. Lastly, the training and validation processes are described. We show how GLR can serve as the training objective, albeit with limitations.

A. METRIC LEARNING ON GRAPHS

Distance metric learning—computation of feature distance between two items endowed with feature vectors—is a popular machine learning sub-topic, encompassing different notions of distance and corresponding algorithms, such as supervised global/local metric learning, unsupervised methods, SVN methods and kernel methods [8]. Notably, for binary classification authors in [9] optimized the *Mahalanobis distance*, defined as

$$d_{i,j} \triangleq \left(\mathbf{f}_i - \mathbf{f}_j\right)^{\top} \mathbf{M} \left(\mathbf{f}_i - \mathbf{f}_j\right), \tag{1}$$

where $\mathbf{M}$ is the PSD metric matrix, and $\mathbf{f}_i, \mathbf{f}_j$ are the feature vectors of data points i, j, respectively. This distance is optimized by minimizing the *large margin nearest neighbors* (LMNN) objective given data. In a nutshell, via metric matrix $\mathbf{M}$, LMNN minimizes the distances between same-label pairs while maximizing the distances between different-label pairs, but no further than a large margin.

Our metric matrix optimization objective is similar to one in [9] in that it defines Mahalanobis distance as the chosen metric for same-/ different-label pairs. However, we adapt the LMNN objective to our constructed graph model instead of a multi-dimensional space. Specifically, in our model datapoints are nodes of a connected graph, and the node labels collectively constitute a *graph signal* on the underlying graph kernel [21], [22]. We call our newly adapted objective *Graph-based Large Margin Nearest Neighbor* (GLMNN).

One key advantage of using a graph in the metric learning context is that we can control the complexity of the training process by strategically choosing *sparse* connections between nodes. For instance, Fig. 1 compares the accuracy and runtime of the model in two cases:
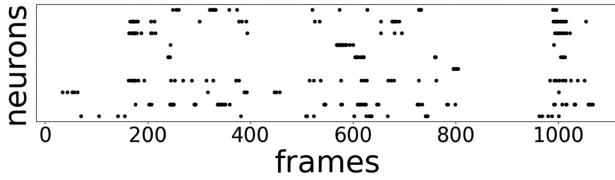
FIGURE 2. Ganglion cells' spikes through time in response to a movie.

when all graph edges are present, and when the maximum degree of nodes is fixed at a constant number. Refer to Section V-A for more information about how the sparse edges are selected. We observe that runtime can be greatly reduced while prediction accuracy is maintained. Each edge in the graph corresponds with one term in the optimization objective. As a result, by strategically removing edges from the graph, we can simplify the objective without significantly degrading performance.

In [23] and [24], the authors trained PSD metric matrix **M** by minimizing the Mahalanobis distance to restore noisy graph signals and classify entries with binary labels, respectively. The training objective is a quantity called *Graph Laplacian Regularizer* (GLR) [7] instead of LMNN. However, for binary classification GLR does not efficiently utilize all available information in the training data; we elaborate this point in the sequel. Lastly, [11] proposed the *Gershgorin Disc Perfect Alignment* (GDPA) linearization technique to reduce the complexity of optimizing a matrix variable with a PSD cone constraint. A faster version of GDPA is also used in [24]. We employ GDPA linearization to speed up the minimization of GLMNN during training.

B. THE DATA

We use the experimental data from [25], where individual ganglion cells' spikes are recorded through time while a naturalistic movie was being played in front of an extracted salamander's retina (see Fig. 2). The movie has 1141 frames, each of size 120×200 pixels. Each frame was displayed for $1/60$ seconds. The time axis was separated into bins of $1/60$ seconds synced with the frames. Each row of the figure shows a specific cell's spikes, where dots represent spikes. The spike signals are binary because a ganglion cell can either fire or stay silent at a given moment; hence, a binary number can describe its state. The brain receives numerous binary signals from many ganglion cells and analyzes all signals together in the process of visual perception. The responses of 113 cells to 297 repeated playbacks of the same movie were recorded. As a result, the whole dataset is stored as two tensors: one tensor describing the stimulus shaped as $1141 \times 120 \times 200$, and one describing the cell responses shaped as $113 \times 297 \times 1141$.

Individual cells fire in only about 3% of the time bins. To create more balance between firings and idleness, we distinguish the "all-silent" state, when no neuron is firing, from any other state when at least one neuron is active. The "all-silent" state frequently appears in retinal recordings, and it is well known in the literature [26]. Hence, we group all ganglion cells so that a time bin's label in the group response is 1 (black) if *any* ganglion cell in a group spikes during that bin, and -1 (white) otherwise. Labels are denoted by $y_i \in \{-1, 1\}$ for $i \in \{1, \dots, 1141\}$. Thus, the group response tensor is shaped as $1 \times 297 \times 1141$.

Further, we consider a batch of U frames immediately preceding the time bin's own frame to predict the label. The number of frames in a batch depends on the utilized feature extraction algorithm. In
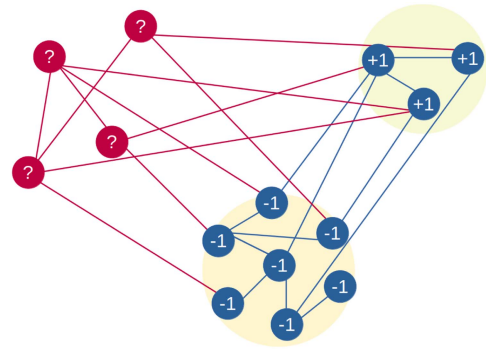

FIGURE 3. Training (blue) and validation (red) nodes in the similarity graph.

our experiments, we used batches of $U = 32$ and $U = 42$ frames. Features for each batch are extracted using methods such as pre-trained CNNs' filters or hand-crafted feature sets like SIFT [27]. The resulting K features extracted from each batch i are denoted by vector $\mathbf{f}_i \in \mathbb{R}^K$. In summary, our dataset consists of $(\mathbf{f}_i, y_i)$ pairs describing a feature vector extracted from a frame batch and its corresponding label. A single $(\mathbf{f}_i, y_i)$ pair is called a datapoint.

C. GRAPH CONSTRUCTION & GLR OPTIMIZATION

First, we split the data into training and validation sets. Datapoints are selected randomly for the sets while ensuring that both sets consist of half 1-labeled datapoints and half -1-labeled datapoints. We then construct a *similarity graph* $\mathcal{G}_t$ with N nodes representing N labeled training datapoints, each with a binary label y_i and a feature vector $\mathbf{f}_i$. We define the weight $w_{i,j}$ of an edge connecting nodes i and j using an exponential kernel and Mahalanobis distance:

$$w_{i,j} = \exp\left(-d_{i,j}\right), \tag{2}$$

where $d_{i,j}$ defined in (1) which depends on $\mathbf{M} \in \mathbb{R}^{K \times K}$. **M** is a *metric matrix* learned through a training process. Moreover, $\mathbf{M} \succeq 0$ is PSD to ensure $d_{i,j} \geq 0$. The edge set $\{w_{i,j}\}$ connecting N nodes thus defines an *adjacency matrix* $\mathbf{W}_t \in \mathbb{R}^{N \times N}$ that specifies graph $\mathcal{G}_t$. The training graph is illustrated in Fig. 3 with blue nodes and edges. The red nodes represent the validation datapoints, which are added to the graph after training. The goal of training is to find an optimal metric matrix **M** such that edges connecting two same-label nodes have noticeably higher weights than edges between opposing-label nodes. If successful, same-label nodes would form local-neighborhood *clusters* as illustrated, and thus the constructed graph becomes amenable to classification.

Let the training objective $Q(\mathbf{M})$ be the GLR (as in [23]) on known label signal $\mathbf{y}_t \in \{-1, 1\}^N$ plus a trace term to ensure that edge weights do not all tend to zero:

$$\min_{\mathbf{M} \succeq 0} \; Q(\mathbf{M}) = \mathbf{y}_t^\top \mathbf{L}_t(\mathbf{M}) \mathbf{y}_t + \mu \operatorname{tr}(\mathbf{M}) \tag{3}$$

where $\mathbf{L}_t \triangleq \operatorname{diag}(\mathbf{W}_t \mathbf{1}) - \mathbf{W}_t$ is the graph Laplacian matrix, $\mu > 0$ is a chosen parameter, $\mathbf{1}$ is an all-one vector of suitable length, and $\operatorname{diag}(\mathbf{v})$ returns a diagonal matrix with diagonal terms $\mathbf{v}$. Note that Laplacian $\mathbf{L}_t$ (by extension graph $\mathcal{G}_t$) is a function of **M**, since **M** defines distances $d_{i,j}$, which in turn determines edge weights $w_{i,j}$ and $\mathbf{W}_t$. The trace term upper bounds the eigenvalues of **M** which,

in turn, ensures that the distance between two finite feature vectors remains finite.

We can expand the objective $Q(\mathbf{M})$ in (3) and substitute the definition of Mahalanobis distance from (1) to arrive at a more clear formulation,

$$\min_{\mathbf{M} \succeq 0} \sum_{(i,j) \in \mathcal{E}} \underbrace{\exp\left(-\left(\mathbf{f}_i - \mathbf{f}_j\right)^\top \mathbf{M}\left(\mathbf{f}_i - \mathbf{f}_j\right)\right)}_{w_{i,j}} \left(y_i - y_j\right)^2 + \mu \operatorname{tr}(\mathbf{M}) \quad (4)$$

where the summation is the connected node-pair expansion of GLR [7] $\mathbf{y}^\top \mathbf{L}(\mathbf{M})\mathbf{y}$, and $\mathbf{y} \in \{-1, 1\}^N$ is the aforementioned known label signal. (We drop the subscript t in the sequel for notational convenience.) When $y_i = y_j$—i.e., nodes i and j have the same labels—each term $w_{i,j}(y_i - y_j)^2 = 0$ regardless of the value of $w_{i,j}$. However, to properly train the metric matrix $\mathbf{M}$, the knowledge that *same-label pairs should be close in feature distance* can be useful; specifically, $d_{i,j} = (\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{M}(\mathbf{f}_i - \mathbf{f}_j) \geq 0$ for same-label pair (i, j) should be small.

D. ESTIMATING LABELS FROM GRAPH

After training, we are left with a similarity graph and an optimized metric matrix. To estimate missing labels, we represent each one of M validation visual events by an additional node (red nodes in Fig. 3). We connect each validation node to training nodes, making sure that at least one training node from each label is connected. Then, we minimize the GLR of this expanded graph $\mathcal{G}$ of $N + M$ nodes:

$$\min_{\mathbf{y}_v} \mathbf{y}^\top \mathbf{L} \mathbf{y}, \quad (5)$$

where $\mathbf{L} \in \mathbb{R}^{(N+M) \times (N+M)}$ is the expanded graph's Laplacian, and $\mathbf{y} = [\mathbf{y}_t^\top \ \mathbf{y}_v^\top]^\top$ consists of both training and validation labels.

After minimization, we hard-threshold the entries of $\mathbf{y}_v^*$ to obtain final visual event label predictions:

$$(\hat{\mathbf{y}}_v)_i = \begin{cases} 1 & \text{if } (\mathbf{y}_v^*)_i > 0 \\ -1 & \text{o.w.} \end{cases}. \quad (6)$$

If $(\hat{y}_v)_i = 1$, then we predict a spike will occur, while if $(\hat{y}_v)_i = -1$, then we predict a spike will not occur.

IV. GLMNN OPTIMIZATION AND GDPA APPROXIMATION

We present a new training objective—a graph adaptation of LMNN—and reformulate the problem as a semi-definite program (SDP). Then, we solve the posed SDP efficiently via GDPA linearization [11].

A. GRAPH-BASED LARGE MARGIN NEAREST NEIGHBOR OBJECTIVE

As previously discussed, when GLR is used as the training objective, same-label node pairs are not used to optimize metric matrix $\mathbf{M}$. However, enforcing large edge weights (small Mahalanobis distances) between same-label node pairs can inform a better similarity graph. This motivates an alternative training objective.

For binary classification, we propose a new objective called *graph-based large margin nearest neighbor* (GLMNN) focusing on same- *and* opposing-label pair distances:

$$\min_{\mathbf{M} \succeq 0} \sum_{(i,j) \in \mathcal{E}|y_i=y_j} d_{i,j}(\mathbf{M})$$

$$+ \rho \sum_{(i,j),(i,l) \in \mathcal{E}|y_i=y_j=-y_l} \left[d_{i,j}(\mathbf{M}) + \gamma - d_{i,l}(\mathbf{M})\right]_+ \quad (7)$$

where $d_{i,j}(\mathbf{M}) = (\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{M}(\mathbf{f}_i - \mathbf{f}_j)$, $\gamma > 0$ is a parameter, and $[a]_+$ returns a if $a \geq 0$ and 0 otherwise. The first sum is an aggregate of distances $d_{i,j}(\mathbf{M})$ between same-label node pairs connected by edges $(i, j) \in \mathcal{E}$. The second sum means that each distance $d_{i,l}(\mathbf{M})$ between opposing-label pair (i, l) would be penalized *only if* it is within γ of a same-label pair distance $d_{i,j}(\mathbf{M})$. Though the idea of trading off same- and opposing-label pair distances originates from LMNN [9], GLMNN adapts LMNN into a graph context, where same- / opposing-label pairs (i, j) are specified as graph nodes i and j connected by edges $\mathcal{E}$ in a graph $\mathcal{G}$. An intentionally sparsely constructed graph $\mathcal{G}$ would thus lead to lower complexity when computing objective (7) due to fewer edges in $\mathcal{E}$.

Towards efficient optimization, we first rewrite the objective (7) as follows. First, we rewrite each distance term $d_{i,j}(\mathbf{M})$ in the two sums as

$$\begin{aligned} d_{i,j}(\mathbf{M}) &= \left(\mathbf{f}_i - \mathbf{f}_j\right)^\top \mathbf{M}\left(\mathbf{f}_i - \mathbf{f}_j\right) \\ &= \operatorname{tr}\left(\left(\mathbf{f}_i - \mathbf{f}_j\right)^\top \mathbf{M}\left(\mathbf{f}_i - \mathbf{f}_j\right)\right) \\ &= \operatorname{tr}\left(\mathbf{M}\left(\mathbf{f}_i - \mathbf{f}_j\right)\left(\mathbf{f}_i - \mathbf{f}_j\right)^\top\right) \\ &= \operatorname{tr}\left(\mathbf{M}\mathbf{F}_{i,j}\right) \end{aligned} \quad (8)$$

where $\mathbf{F}_{i,j} \triangleq (\mathbf{f}_i - \mathbf{f}_j)(\mathbf{f}_i - \mathbf{f}_j)^\top$. Second, to linearize the non-negativity function $[\mathbf{a}]_+$, we define non-negative auxiliary variable $\delta_{i,j,l}$ that is an upper bound on $d_{i,j}(\mathbf{M}) + \gamma - d_{i,l}(\mathbf{M})$. Hence, minimizing $\delta_{i,j,l}$ is equivalent to minimizing $[d_{i,j}(\mathbf{M}) + \gamma - d_{i,l}(\mathbf{M})]_+$. The optimization (7) can now be rewritten as

$$\min_{\{\delta_{i,j,l}\}, \mathbf{M} \succeq 0} \sum_{(i,j) \in \mathcal{E}|y_i=y_j} \operatorname{tr}\left(\mathbf{M}\mathbf{F}_{i,j}\right) + \rho \sum_{(i,j),(i,l) \in \mathcal{E}|y_i=y_j=-y_l} \delta_{i,j,l}$$

$$\text{s.t. } \delta_{i,j,l} \geq \operatorname{tr}\left(\mathbf{M}\mathbf{F}_{i,j}\right) + \gamma - \operatorname{tr}\left(\mathbf{M}\mathbf{F}_{i,l}\right)$$

$$\delta_{i,j,l} \geq 0 \quad (9)$$

where the variables are $\{\delta_{i,j,l}\}$ and $\mathbf{M}$. The problem is a *semi-definite programming* (SDP) problem with linear objective and linear constraints plus a PSD cone constraint $\mathbf{M} \succeq 0$ [10].

While SDP problems can be solved in polynomial time using an off-the-shelf SDP solver, the worst-case complexity $\mathcal{O}(K^3)$ is still high. Next, we adopt a fast approximation algorithm that addresses convex optimization problems with a PSD cone constraint called *GDPA linearization* [11].

B. GDPA LINEARIZATION ALGORITHM

1) REPLACING PSD CONE CONSTRAINT

The basic idea of GDPA linearization is to replace the PSD cone constraint with a set of linear constraints per iteration, so that together with the linear objective and other linear constraints, the SDP problem simplifies to a *linear program* (LP), solvable using a state-of-the-art LP solver[1] per iteration. GDPA linearization is based on a well-known linear algebra theorem called the *Gershgorin Circle Theorem* (GCT) [29], a version of which states that each real eigenvalue λ of a real symmetric matrix $\mathbf{M}$ resides inside at least one *Gershgorin disc* Ψ_i corresponding to row i, with center $c_i \triangleq M_{i,i}$ and radius $r_i \triangleq \sum_{j \neq i} |M_{i,j}|$, i.e., $\exists i$, s.t. $c_i - r_i \leq \lambda \leq c_i + r_i$. The corollary is that the smallest eigenvalue $\lambda_{\min}$ is lower-bounded by the smallest

---

[1]A representative state-of-the-art general LP solver is [28], which has complexity $\mathcal{O}(K^{2.055})$.

disc left-end $\lambda_{\min}^-$, i.e.,

$$\lambda_{\min}^-(\mathbf{M}) \triangleq \min_i M_{i,i} - \sum_{j \neq i} |M_{i,j}| \leq \lambda_{\min}(\mathbf{M}). \quad (10)$$

Thus, to ensure $\mathbf{M} \succeq 0$, one can enforce linear constraints $\lambda_{\min}^-(\mathbf{M}) \geq 0$, i.e., all disc left-ends are at least zero.

However, because the lower bound $\lambda_{\min}^-(\mathbf{M}) \leq \lambda_{\min}(\mathbf{M})$ is often loose, enforcing $\lambda_{\min}^-(\mathbf{M}) \geq 0$ directly would mean we are overly restricting the search space, not allowing feasible solutions $\mathbf{M}'$ where $\lambda_{\min}^-(\mathbf{M}') < 0 \leq \lambda_{\min}(\mathbf{M}')$.

Knowing that a *similarity transform* $\mathbf{SMS}^{-1}$ of $\mathbf{M}$ shares the same set of eigenvalues (assuming $\mathbf{S}$ is invertible), we first replace constraint $\lambda_{\min}^-(\mathbf{M}) \geq 0$ with $\lambda_{\min}^-(\mathbf{SMS}^{-1}) \geq 0$. Assuming a diagonal $\mathbf{S}$, the N disc left-end constraints are

$$M_{i,i} - \sum_{j | j \neq i} \left| \frac{s_i M_{i,j}}{s_j} \right| \geq 0, \ \forall i. \quad (11)$$

Second, we can select diagonal $\mathbf{S}$ so that the lower-bound gap can be eliminated for a given $\mathbf{M}$, i.e., $\lambda_{\min}^-(\mathbf{SMS}^{-1}) = \lambda_{\min}(\mathbf{SMS}^{-1}) = \lambda_{\min}(\mathbf{M})$, as follows. First, we recall that a recent linear algebra theorem called *Gershgorin Disc Perfect Alignment* (GDPA) [11] states that $\lambda_{\min}^-(\mathbf{SMS}^{-1}) = \lambda_{\min}(\mathbf{M})$ for a diagonal $\mathbf{S}$ if

1) $s_i = 1/v_i$, $\forall i$ and $\mathbf{v}$ is the first eigenvector of $\mathbf{M}$, and
2) $\mathbf{M}$ is a Laplacian matrix to a *balanced* signed graph.

By the *Cartwright-Harary Theorem* (CHT) [30], a signed graph (i.e., a graph with both positive and negative edges) is balanced if and only if nodes can be colored into two colors (e.g., blue and red), such that positive / negative edges connect nodes of the same / different colors. Given GDPA, we can design a fast approximation algorithm to (9) as follows.

2) ALGORITHM DESIGN

Assume that at iteration $\tau - 1$, a feasible solution $\mathbf{M}^\tau \succeq 0$ to (9) has been computed, where $\mathbf{M}^\tau$ is a Laplacian to a balanced signed graph $\mathcal{G}^s$, i.e., nodes in $\mathcal{G}^s$ can be assigned colors blue and red, so that edges connecting same- / different-color pairs (i, j) have positive / negative weights $w_{i,j}$ (denoted by $-M_{i,j}^\tau$, since by definition Laplacian $\mathbf{M}^\tau \triangleq \mathbf{D}^\tau - \mathbf{W}^\tau + \mathrm{diag}(\mathbf{W}^\tau)$). We first compute first eigenvector $\mathbf{v}$ for $\mathbf{M}^\tau$ using a fast extreme eigenvector computation algorithm such as LOBPCG [31], using which we define $s_i = 1/v_i$, $\forall i$. Linear constraints (11) that ensure solution $\mathbf{M} \succeq 0$ are now well defined.

At iteration τ, we optimize diagonal terms $\{M_{l,l}\}_{\forall l}$ plus one row / column i of $\mathbf{M}$ at a time: for each row / column i, we optimize $\{M_{i,j}, M_{j,i}\}_{\forall j \neq i}$ twice, each time assuming node i is colored either red or blue. The assumed color implies positive / negative signs for $M_{i,j}$'s and $M_{j,i}$'s to maintain graph balance to other colored nodes $j \neq i$, which we can enforce using additional linear constraints in a LP. To illustrate, denote the set of all blue nodes of $\mathcal{G}^s$ at iteration τ by $\mathcal{N}_{b,\tau}$ and the set of all red nodes of $\mathcal{G}^s$ at iteration τ by $\mathcal{N}_{r,\tau}$. Then, if it is assumed that $i \in \mathcal{N}_{b,\tau}$, the following linear constraints are imposed,

$$\begin{cases} M_{i,j} = M_{j,i} \leq 0 & \text{if } j \in \mathcal{N}_{b,\tau} \\ M_{i,j} = M_{j,i} \geq 0 & \text{if } j \in \mathcal{N}_{r,\tau} \end{cases} \quad (12)$$

Conversely, if it is assumed that $i \in \mathcal{N}_{r,\tau}$, the following linear constraints are enforced instead,

$$\begin{cases} M_{i,j} = M_{j,i} \leq 0 & \text{if } j \in \mathcal{N}_{r,\tau} \\ M_{i,j} = M_{j,i} \geq 0 & \text{if } j \in \mathcal{N}_{b,\tau} \end{cases} \quad (13)$$

We select the smaller of the two objective values corresponding to the two assumed colors as the color for node i. We optimize all rows / columns i in this manner in turn till convergence, resulting in optimal solution $\mathbf{M}^{\tau+1}$ for iteration τ. A new first eigenvector $\mathbf{v}$ for obtained solution $\mathbf{M}^{\tau+1}$ is computed again using LOBPCG, and $\{s_i\}$ for linear constraints (11) for the next iteration are defined. The algorithm continues till solution $\mathbf{M}^\tau$ converges.

V. EXPERIMENTS
A. EXPERIMENT SETUP

We first extracted visual features from a target fish movie [25] using three methods. The first is the output of the first layer of a pre-trained CNN called "slowfast-r50" [32], [33]. The second is the output of the second layer of another pre-trained CNN called "SOE-Net" [34], [35]. The Slowfast network was trained on an action recognition task in videos while SOE-Net was developed for simultaneous audio-video texture analysis. The third is a 3D version of the well-known *scale-invariant feature transform* (SIFT) [36], [37]. The Slowfast and 3D-SIFT algorithms both use batches of 32 frames while SOE-Net uses batches of 42 frames. In addition, we used SIFT features extracted from the MNIST dataset [38], consisted of images of handwritten digits from 0 to 9 and their labels. Since the neuron dataset contains binary labels (-1 and $+1$), we separated only two digits of the MNIST set (0 and 1) to better match our target dataset.

We compared our algorithm against five representative state-of-the-art classification schemes: CNN as implemented in [3], [4], [5], RNN similar to [6], XGBoost [39] (a Python implementation of gradient boosting decision trees (GBDT) [40]), logistic regression (LR) [41], and K nearest neighbors (kNN) [42]. All algorithms were trained on the same features.

After feature extraction, we connected the training and validation nodes as follows. For each set, we randomly selected half of the nodes from the 1-labeled bins and the other half from -1-labeled nodes; this ensured that the data sets were evenly distributed between 1 and -1 labels. Edges between two training nodes or two validation nodes were constructed based on their temporal proximity; for example, to connect a node corresponding to time interval t to five other nodes, we chose nodes corresponding to time intervals closest to t, three of them before t and two after t. Edges between a training node and a validation node were selected with a similar strategy; the only difference was that we ensured half the connected nodes were labeled 1 and half are labeled -1. The maximum number of edges between two training nodes was set at parameter D_t, between two validation nodes was set at D_v, and between a training and a validation node was set at D_{vt}. Thus, the maximum degree of a training node was $D_t + D_{vt}$, and the maximum degree of a validation node was $D_v + 2D_{vt}$.

After constructing a graph connecting all training and validation nodes, the metric matrix was optimized on the graph as described in Section IV. We tested three metric matrix optimization algorithms. The first was the GLR objective (3) optimized via gradient descent. The second was the GLMNN objective (9) optimized via an off-the-shelf SDP solver called SeDuMi [43]. The third optimized the same GLMNN objective (9) via our proposed GDPA linearization algorithm described in Section IV-B. To randomize our experiments, we repeated each experiment with P_t different training sets and P_v different validation sets, meaning that each point in the following plots is the average result of $P_t P_v$ trials. The values of P_t and P_v are reported on the y-axis of each plot.
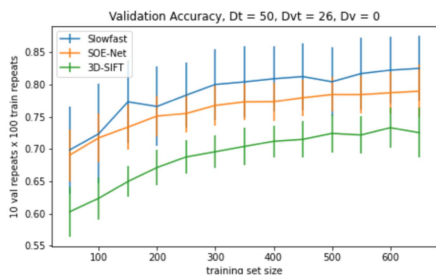
FIGURE 4. Performance of GLR minimization with three different feature extraction algorithms.
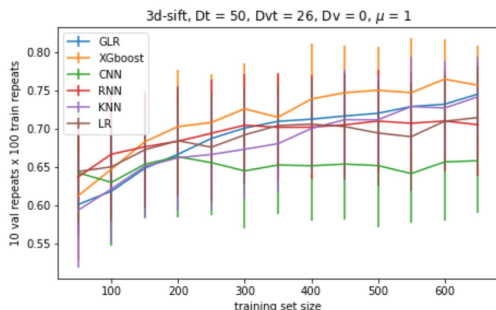

FIGURE 5. Validation accuracy of GLR compared to the state-of-the-art: XGBoost [39], convolutional neural network (CNN) [3], [4], [5], recurrent neural network (RNN) [6], K-Nearest neighbors (KNN) [42], and logistic regression (LR) [41].

B. EXPERIMENT RESULTS
An important advantage of a graph-based classifier is the use of computed similarity edges connecting validation nodes based on available per-node features to estimate their labels, instead of computing labels for each validation node individually. This property is important for semi-supervised learning, where the training set is small compared to the test set. We focus on this setting in our experiments.

1) INTERPRETABILITY VS. PERFORMANCE IN FEATURES
The validation accuracy obtained by minimizing the GLR objective (3) are shown in Fig. 4. Each curve illustrates the performance using one of the three feature extraction algorithms described in Section V-A, all extracted from the fish movie [25]. We observe that Slowfast and SOE-Net achieved a higher accuracy than 3D-SIFT; however, they are less interpretable since their filters are optimized in a purely data-driven manner without clear interpretation. This illustrates one tradeoff between interpretability and performance.

2) PREDICTION ACCURACY
The prediction accuracy of our graph-based classifier is analyzed jointly by Figs. 5 and 6. We compared GLR against four other methods in Fig. 5. We observe that GLR's performance is on par with the benchmarks. This shows that a graph-based classifier inherently suffers no performance disadvantage against its competitors.

As discussed in Section IV, GLMNN utilizes features of same-label pairs to further optimize the metric matrix $\mathbf{M}$. However, minimizing GLMNN using an off-the-shelf SDP solver such as SeDuMi is time-consuming. Thus, we tested SeDuMi only for smaller
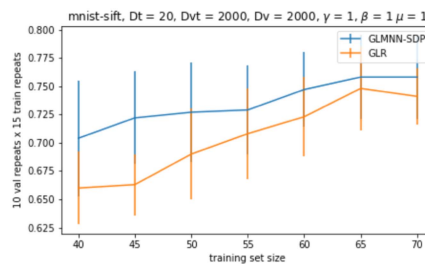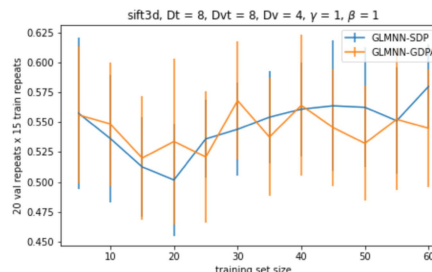

FIGURE 6. Validation accuracy of GLMNN vs. GLR.

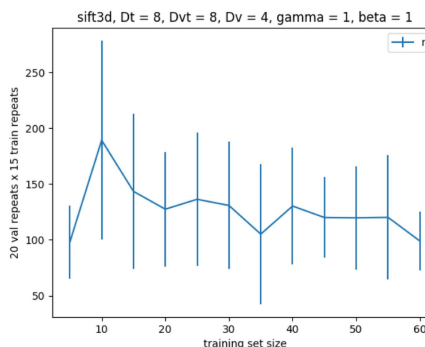**FIGURE 7.** Comparison between obtained validation accuracies by SDP and GDPA.

FIGURE 8. Number of times a linear program was run during GDPA optimization.

datasets than in Fig. 5. A comparison is made between GLMNN minimization using SeDuMi and GLR in Fig. 6. We see that GLMNN noticeably outperformed GLR in classification performance. Further, we observe that the difference between GLMNN and GLR decreased as the training set became larger. One explanation is that the extra information (i.e., distances between same-label pairs) used in GLMNN is more critical for smaller sets where the training data does not provide sufficient information.

3) SPEED
To speed up GLMNN optimization, GDPA linearization algorithm was used. The effectiveness of GDPA is illustrated in Figs. 7 and 8. Fig. 7 compares the obtained accuracy by GLMNN minimization with an SDP solver and GDPA linearization. We observe that, though GDPA provided only approximate solutions, its performance was reasonably close to an SDP solver as shown in Fig. 7.

On the other hand, GDPA greatly reduces algorithm runtime for large datasets. A general SDP solver has time complexity $\mathcal{O}(K^3)$. GDPA approximates SDP via a number of iterative linear programs.
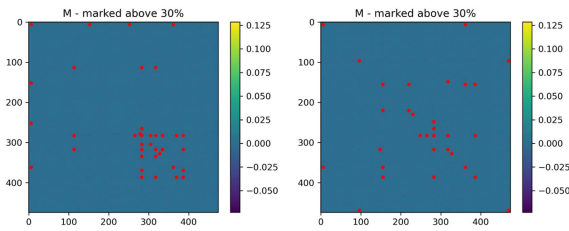
FIGURE 9. Optimized metric matrix for various training sets. Red dots show entries with larger values than 30% of the maximum.
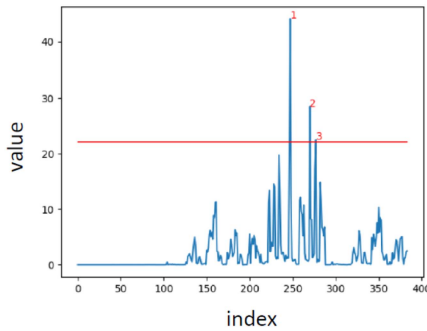

FIGURE 10. Diagonal elements of an optimized metric matrix.

Each linear program requires roughly $\mathcal{O}(K^2)$ to compute. As observed in Fig. 8, the number of required linear programs in GDPA till convergence stays roughly constant. Thus, GDPA reduces time complexity to $\mathcal{O}(K^2)$. The precision of the GDPA algorithm is controlled by the definition of convergence.

4) INTERPRETABILITY

The optimized metric matrix $\mathbf{M}^*$ enables a level of interpretability by depicting the relevance of features to the classification. Diagonal elements of $\mathbf{M}^*$ show the singular contribution of each feature to the distance, while off-diagonal elements show the relevance of features to each other. It is observed in our experiments that $\mathbf{M}^*$ tends to have a few large elements, while all the other elements are small. This means that only a few features contribute substantially to the classification task, which provides a way for feature selection.

Fig. 9 shows visualizations of two example metric matrices in our method for different training sets. All entries larger than 30% of the maximum entry are marked by a red dot. Since the training sets are randomly constructed, different combinations of training nodes lead to different optimized metric matrices. We observe that only a few entries (features) along the diagonal contributed substantially to label prediction.

These dominant features can be extracted and visualized. For example, Fig. 10 shows the diagonal entries of an optimized metric matrix using 3D-SIFT features extracted from the fish movie. We can separate the dominant features via thresholding. The red line in Fig. 10 shows an example of a threshold set at 50% of the maximum entry. How these relevant features are correlated to each other are encoded in off-diagonal terms.

The interpretability of the features themselves varies among different algorithms. For example, 3D-SIFT is more interpretable than CNN-based features like Slowfast and SOE-Net. Each element of the feature vector obtained by 3D-SIFT unambiguously corresponds to the gradients of a specific area of the frame batch in a specific

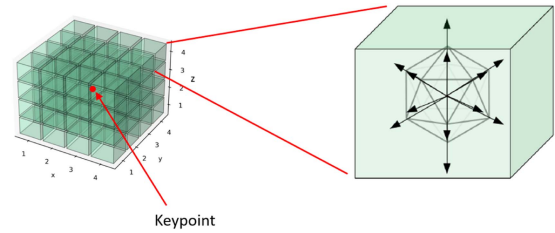**FIGURE 11.** 3D-SIFT keypoints on the fish movie frames.

FIGURE 12. Subregions around the 3D-SIFT keypoint and the icosahedron whose vertices correspond to histogram bins created in each subregion.
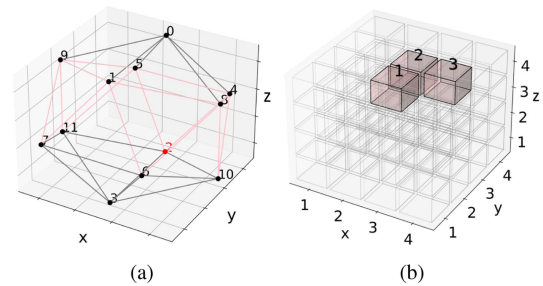

FIGURE 13. (a) Each element of a 3D-SIFT histogram corresponds to one of the 12 vertices of a regular icosahedron. (b) Locations of the most dominant features relative to the keypoint for 3D-SIFT.

direction. This area is determined by the 3D-SIFT keypoint and the location of each element relative to the keypoint.

There may be any number of 3D-SIFT keypoints found in a frame batch. Nevertheless, all feature vectors should have the same length in our model; therefore, we need to select a fixed number of keypoints in each frame batch and disregard the others. Many keypoint selection strategies can be adopted. One example, which is employed in our experiments, is selecting the keypoint closest to the center of the frame batch. To illustrate, Fig. 11 shows the selected 3D-SIFT keypoint on several frame batches from the fish movie. Selecting the most central keypoint is a rudimentary provision to cover the largest possible area by the keypoint's descriptor.

The keypoint descriptor is computed as follows. A neighborhood around the keypoint is considered and divided into 64 subregions. Local gradients are computed in each subregion and a histogram is constructed which shows the cumulative projections of the gradients in 12 directions in the 3D space. Each direction corresponds to a vertex of a regular icosahedron. To compute the cumulative gradient projections in each direction, only those local gradients are considered that intersect with at least one of the faces comprising the vertex in question. Fig. 12 shows the 64 subregions and the icosahedron in one of the subregions.

The subregions are numbered based on their Cartesian coordinates with respect to the keypoint. Similarly, the vertices of the icosahedron are also numbered as in Fig. 13(a). The 64 histograms, each consisting of 12 bins, are concatenated to form the keypoint descriptor which we assign to the frame batch as its feature vector.
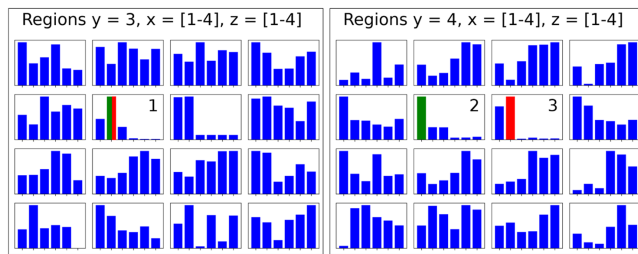
FIGURE 14. Diagonal elements of M* visualized as histograms inside subregions around the 3D-SIFT keypoint. Subregions labeled 1, 2 and 3 are the same subregions as in Fig. 13(b). Each bin in a histogram represents a specific direction in 3D space.

In other words, the feature vector has $12 \times 64 = 768$ dimensions. Consequently, each element of a 3D-SIFT feature vector can be described by two numbers: one number determining its corresponding subregion relative to the keypoint, and one number depicting its corresponding vertex in the icosahedron (i.e. the general direction of the gradients).

Given this knowledge of what each element in a 3D-SIFT feature vector means, we can generate more detailed interpretations of the prediction process. For instance, using the diagonal elements of M* from Fig. 10, we can illustrate the locations of dominant features relative to the keypoint as in Fig. 13(b).

Furthermore, we can describe the dominant features with their direction and location relative to the keypoint. For example, we can describe the most dominant feature in Fig. 10 as "the cumulative projections of local gradients in the general direction of ($\theta = 148.28^o$, $\phi = 90^o$) in spherical coordinates, computed in a subregion above the keypoint and to its northwest." This long description can be summarized by two numbers: (2, 41). The first number (i.e. 2) notes the corresponding icosahedron vertex of the most dominant feature. This vertex is marked in red in Fig. 13(a). The second number (i.e. 41) depicts the corresponding subregion (i.e. location) of the most dominant feature. This subregion is marked by number 1 and pink color in Fig. 13(b). The axes of Fig. 13 are defined based on the keypoint's orientation which is known.

The diagonal elements of M* can be visualized as histograms in 64 subregions around the 3D-SIFT keypoint. Fig. 14 illustrates 32 of such histograms with the following coordinates $x = \{1, 2, 3, 4\}$, $y = \{3, 4\}$, $z = \{1, 2, 3, 4\}$. We can locate the most dominant feature in Fig. 10 on this illustration as well; it corresponds to the 2nd bin of the 42nd histogram. This feature is distinguished by the color half-red half-green in Fig. 14. Each bin in a histogram corresponds to an icosahedron vertex. The histograms in Fig. 14 have 6 bins instead of 12; the reason is that the 3D-SIFT feature vectors were subsampled with a rate of 2 : 1 to cut the time complexity into a quarter of its original value.

In addition to single features, we can illustrate *pairs* of features whose corresponding non-diagonal element in M* have the largest values. Two such pairs are illustrated in Fig. 14 with red and green colors. The red pair correspond to the highest non-diagonal element, and the green pair correspond to the second highest. A large non-diagonal element of M* means that the correlation between the two features is highly informative for label prediction.

VI. CONCLUSION
To extract knowledge from ganglion cell responses to visual stimuli, we propose an interpretable graph-based binary classifier learning

framework, where a positive semi-definite (PSD) metric matrix M is optimized to determine feature distances between nodes representing visual events. Using a new objective called graph-based large margin nearest neighbor (GLMNN), we formulate a semi-definite programming (SDP) problem to optimize M, which is efficiently approximated via an adoption of Gershgorin disc perfect alignment (GDPA) linearization. Given an optimized metric matrix M, one can directly identify features substantially contributed to label classification by locating large-magnitude diagonal terms. Experimental results show that our graph-based classifier is competitive with state-of-the-art classification schemes, while offering a level of interpretability.

For future work, our model can be employed to predict the responses of individual ganglion cells, various physiological groups of ganglion cells, or even retinal responses of other organisms such as mice and primates [44], [45]. Analyzing the optimized model on these datasets can reveal further insight into the retinal computation.

REFERENCES
[1] L. A. Remington, *Clinical Anatomy of the Visual System*, 2nd ed., Amsterdam, The Netherlands: Elsevier, 2005.
[2] T. Gollisch and M. Meister, "Eye smarter than scientists believed: Neural computations in circuits of the retina," *Neuron*, vol. 65, no. 2, pp. 150–164, 2010.
[3] L. McIntosh, N. Maheswaranathan, A. Nayebi, S. Ganguli, and S. Baccus, "Deep learning models of the retinal response to natural scenes," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1369–1377.
[4] A. Lozano, C. Soto-Sanchez, J. Garrigos, J. J. Martinez, J. M. Ferrández, and E. Fernandez, "A 3D convolutional neural network to model retinal ganglion cell's responses to light patterns in mice," *Int. J. Neural Syst.*, vol. 28, no. 10, 2018, Art. no. 1850043.
[5] H. Tanaka, A. Nayebi, N. Maheswaranathan, L. McIntosh, S. Baccus, and S. Ganguli, "From deep learning to mechanistic understanding in neuroscience: The structure of retinal prediction," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8537–8547.
[6] E. Batty et al., "Multilayer recurrent network models of primate retinal ganglion cell responses," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 2174–2186.
[7] J. Pang and G. Cheung, "Graph Laplacian regularization for image denoising: Analysis in the continuous domain," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1770–1785, Apr. 2017.
[8] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," *Michigan State Universiy*, vol. 2, no. 2, 2006, Art. no. 4.
[9] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, no. 2, pp. 207–244, 2009.
[10] Z.-q. Luo, W.-k. Ma, A. M.-c. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 20–34, May 2010.
[11] C. Yang, G. Cheung, and W. Hu, "Signed graph metric learning via Gershgorin disc perfect alignment," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 7219–7234, Oct. 2022.
[12] D. Marshak, "Retinal ganglion cells: Anatomy," in *Encyclopedia of Neuroscience*. Oxford, U.K.: Academic Press, 2009, pp. 211–218.
[13] G. Zeck, M. Bethge, and J. H. Macke, "Receptive fields without spike-triggering," in *Advances in Neural Information Processing Systems*, Red Hook, NY, USA: Curran Associates Inc., 2007.
[14] J. B. Demb, K. Zaghloul, L. Haarsma, and P. Sterling, "Bipolar cells contribute to nonlinear spatial summation in the brisk-transient (Y) ganglion cell in mammalian retina," *J. Neurosci.*, vol. 21, no. 19, pp. 7447–7454, 2001.
[15] M. J. Berry, I. H. Brivanlou, T. A. Jordan, and M. Meister, "Anticipation of moving stimuli by the retina," *Nature*, vol. 398, no. 6725, pp. 334–338, 1999.

[16] B. P. Ölveczky, S. A. Baccus, and M. Meister, "Segregation of object and background motion in the retina," *Nature*, vol. 423, no. 6938, pp. 401–408, 2003.

[17] T. A. Münch, R. A. D. Silveira, S. Siegert, T. J. Viney, G. B. Awatramani, and B. Roska, "Approach sensitivity in the retina processed by a multifunctional neural circuit," *Nature Neurosci.*, vol. 12, no. 10, pp. 1308–1316, 2009.

[18] E. Chichilnisky, "A simple white noise analysis of neuronal light responses," *Netw.: Computation Neural Syst.*, vol. 12, no. 2, 2001, Art. no. 199.

[19] S. V. David, W. E. Vinje, and J. L. Gallant, "Natural stimulus statistics alter the receptive field structure of v1 neurons," *J. Neurosci.*, vol. 24, no. 31, pp. 6991–7006, 2004.

[20] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, 2020, Art. no. 18.

[21] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[22] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, "Graph spectral image processing," *Proc. IEEE*, vol. 106, no. 5, pp. 907–930, May 2018.

[23] W. Hu, X. Gao, G. Cheung, and Z. Guo, "Feature graph learning for 3D point cloud denoising," *IEEE Trans. Signal Process.*, vol. 68, pp. 2841–2856, 2020.

[24] X. Li, C. Yang, W. Tong, F. Shi, and G. Zhai, "Fast graph-based binary classifier learning via further relaxation of semi-definite relaxation," in *Proc. 5th Int. Conf. Image Graph. Process.*, 2022, pp. 89–95.

[25] O. Marre, G. Tkacik, D. Amodei, E. Schneidman, W. Bialek, and M. Berry, "Multi-electrode array recording from salamander retinal ganglion cells," *IST Austria*, 2017, doi: 10.15479/AT:ISTA:61.

[26] G. Tkavcik, O. Marre, D. Amodei, E. Schneidman, W. Bialek, and M. J. Berry, "Searching for collective behavior in a large network of sensory neurons," *PLoS Comput. Biol.*, vol. 10, no. 1, 2014, Art. no. e1003408.

[27] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.

[28] S. Jiang, Z. Song, O. Weinstein, and H. Zhang, "A faster algorithm for solving general LPs," in *Proc. 53rd Annu. ACM SIGACT Symp. Theory Comput.*, 2021, pp. 823–832.

[29] R. S. Varga, *Gervgorin and His Circles*, vol. 36. Berlin, Germany: Springer, 2010.

[30] D. Cartwright and F. Harary, "Structural balance: A generalization of Heider's theory," *Psychol. Rev.*, vol. 63, no. 5, pp. 277–293, 1956.

[31] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM J. Sci. Comput.*, vol. 23, no. 2, pp. 517–541, 2001.

[32] H. T. Shen et al., "PyTorchVideo: A deep learning library for video understanding," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 3783–3786. [Online]. Available: https://pytorchvideo.org/

[33] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6202–6211.

[34] I. Hadji and R. P. Wildes, "A spatiotemporal oriented energy network for dynamic texture recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3066–3074.

[35] I. Hadji, "SOE-Net," 2020. Accessed: Apr. 9, 2023. [Online]. Available: https://github.com/hadjisma/SOE-Net

[36] B. Rister, M. A. Horowitz, and D. L. Rubin, "Volumetric image registration from invariant keypoints," *IEEE Trans. Image Process.*, vol. 26, no. 10, pp. 4900–4910, Oct. 2017.

[37] B. Rister, "SIFT3D," 2019. [Online]. Available: https://github.com/bbrister/SIFT3D

[38] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[39] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.

[40] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, pp. 1189–1232, 2001.

[41] D. W. Hosmer Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. 398. Hoboken, NJ, USA: Wiley, 2013.

[42] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[43] J. Strum, "A matlab toolbox for optimization over symmetric cones," *Optim. Methods Softw.*, vol. 11, pp. 625–653, 2001.

[44] J. Goetz et al., "Unified classification of mouse retinal ganglion cells using function, morphology, and gene expression," *Cell Rep.*, vol. 40, no. 2, 2022, Art. no. 111040. [Online]. Available: http://rgctypes.org/

[45] J. Shlens et al., "The structure of multi-neuron firing patterns in primate retina," *J. Neurosci.*, vol. 26, no. 32, pp. 8254–8266, 2006.