Interpretable Neuron Structuring with Graph Spectral Regularization

Alexander Tong[1], David van Dijk[2], Jay S. Stanley III[2], Matthew Amodio[1], Kristina Yim[2], Rebecca Muhle[2], James Noonan[2], Guy Wolf[3], and Smita Krishnaswamy[1,2](✉)

[1] Yale Department of Computer Science, New Haven, USA
smita.krishnaswamy@yale.edu
[2] Yale Department of Genetics, New Haven, USA
[3] Department of Mathematics and Statistics,
Université de Montréal, Mila, Montreal, Canada

Abstract. While neural networks are powerful approximators used to classify or embed data into lower dimensional spaces, they are often regarded as black boxes with uninterpretable features. Here we propose *Graph Spectral Regularization* for making hidden layers more interpretable without significantly impacting performance on the primary task. Taking inspiration from spatial organization and localization of neuron activations in biological networks, we use a graph Laplacian penalty to structure the activations within a layer. This penalty encourages activations to be smooth either on a predetermined graph or on a feature-space graph learned from the data via co-activations of a hidden layer of the neural network. We show numerous uses for this additional structure including cluster indication and visualization in biological and image data sets.

Keywords: Neural Network Interpretability · Graph learning · Feature saliency

1 Introduction

Common intuitions and motivating explanations for the success of deep learning approaches rely on analogies between artificial and biological neural networks, and the mechanism they use for processing information. However, one aspect that is overlooked is the spatial organization of neurons in the brain. Indeed, the hierarchical spatial organization of neurons, determined via fMRI and other technologies [13,16], is often leveraged in neuroscience works to explore, understand, and interpret various neural processing mechanisms and high-level brain functions. In artificial neural networks (ANN), on the other hand, hidden layers offer no organization that can be regarded as equivalent to the biological one. This lack of organization poses great difficulties in exploring and interpreting

A. Tong, D. Dijk, G. Wolf and S. Krishnaswamy—Equal contribution.

M. R. Berthold et al. (Eds.): IDA 2020, LNCS 12080, pp. 509–521, 2020.
https://doi.org/10.1007/978-3-030-44584-3_40

the internal data representations provided by hidden layers of ANNs and the information encoded by them. This challenge, in turn, gives rise to the common treatment of ANNs as black boxes whose operation and data processing mechanisms cannot be easily understood. To address this issue, we focus on the problem of modifying ANNs to learn more interpretable feature spaces without degrading their primary task performance.

While most neural networks are treated as black boxes, we note that there are methods in ANN literature for understanding the activations of filters in convolutional neural networks (CNNs) [11], either by examining trained networks [24], or by learning a better representation [12,17,18,22,25], but such methods rarely apply to other types of networks, in particular dense neural networks (DNNs) where a single activation is often not interpretable on its own. Furthermore, convolutions only apply to datatypes where we know the feature structure apriori, as in the case of images and natural language. In layers of a DNN, there is no enforced structure between neurons. The correspondence between neurons and concepts is only determined based on the random initialization of the network. In this work, we encourage *structure between neurons* in the same layer, creating more localized and interpretable layers in dense architectures.

More specifically we propose a *Graph Spectral Regularization* to encourage arbitrary graph structure between neurons within a layer. The internal layers of a neural network are constrained to take the structure of a graph, with graph neighbors activating on similar inputs. This allows us to map the activations of a given layer over the graph and interpret new input by examining the activations. We show that graph-structuring a hidden layer causes useful, interpretable features to emerge. For instance, we show that grid-structuring a layer of a classification network creates a structure over which convolution can be applied, and local receptive fields can be traced to understand classification decisions.

While a majority of the time imposing a known graph structure gives interpretable results, there are circumstances where we would like to learn the graph structure from data. In such cases we can learn and emphasize the natural graph structure of the feature space. We do this by an iterative process of encoding the data, and modifying the graph based on the feature co-activation patterns. This procedure reinforces existing patterns in the data. This allows us to learn an abstracted graph structure of features in high-dimensional domains such as single-cell RNA sequencing.

The main contributions of this work are as follows: (1) Demonstration of hierarchical, spatial, and smoothed feature maps for interpretability in dense networks. (2) A novel method for learning and reinforcing the natural graph structure for complex feature spaces. (3) Demonstration of graph learning and abstraction on single-cell RNA-sequencing data.

2 Related Work

Disentangled Representation Learning: While there is no precise definition of what makes for a disentangled representation, the aim is to learn a representation that axis aligns with the generative factors of the data [2,8]. [9] suggest a way to disentangle the representation of variational autoencoders [10] with β-VAE. Subsequent work has generalized this to discrete representations [5], and simple hierarchical representations [6]. These works focus on learning a single vector representation of the data, where each element represents a single concept. In contrast, our work learns a representation where groups of neurons may be involved in representing a single concept. Moreover, disentangled representation learning can only be applied to unsupervised models and only the most compressed level of either an autoencoder [9] or generative adversarial network as in [4], whereas graph spectral regularization (GSR) can be applied to any or all layers of the network.

Graph Structure in ANNs: Graph based penalties have been used in the graph signal processing literature [3,21,26], but are rarely used in an ANN setting. In the biological data setting, [14] used a graph penalty in sparse logistic regression on gene expression data. Another way of utilizing graph structure is through graph convolutional networks (GCN). GCNs are a related body of work introduced by [7], and expanded on by [19], but focus on a different set of problems (For an overview see [23]). GCNs require a known graph structure. We focus on learning a graph representation of general data. This learned graph representation could be used as the input to a GCN similar to our MNIST example.

3 Enforcing Graph Structure

We consider the intra-layer relationships between neurons or larger structures such as capsules. For a given layer of neurons we construct a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_N\}$ the set of vertices and $E \subseteq V \times V$ the set of edges. Let W be the weighted symmetric adjacency matrix of size $N \times N$ with $W_{ij} = W_{ji} \geq 0$ representing the weight of the edge between v_i and v_j. The graph Laplacian L is then defined as $L = D - W$ where $D_{ii} = \sum_j W_{ij}$ and $D_{ij} = 0$ for $i \neq j$.

To enforce smoothing we use the Laplacian smoothing loss. On some activation vector z and fixed Laplacian L we formulate the graph spectral regularization function G as:

$$G(z, \mathbf{L}) = z^T \mathbf{L} z = \sum_{ij} W_{ij} ||z_i - z_j|| \tag{1}$$

where $|| \cdot ||$ denotes the Frobenius norm. We add it to the reconstruction or classification loss with a weighting term α. This adds an additional objective that activations should be smooth along the graph defined by L. This optimization procedure applies to any multi-layer model and valid graph Laplacian. We apply this algorithm to grid, and hierarchical graph structures on both autoencoder and classification dense architectures.

Algorithm 1. Graph Learning

```
Input batches $x_i$, model $M$ with latent layer activations $z_i$, regularization weight $\alpha$.
Pre-train $M$ on $x_i$ with $\alpha = 0$
for $i = 1$ to $T$ do
    Create Graph Laplacian $L_i$ from activations $z_i$
    for $j = 1$ to $m$ do
        Train $M$ on $x_i$ with $\alpha = w$ and $L = L_i$ with MSE + loss in eq. 1
    end for
end for
```

3.1 Learning and Reinforcing an Abstracted Feature-Space Graph

Instead of enforcing smoothness over a fixed graph, we can learn a feature graph from the data (See Algorithm 1) using neural network activations themselves to bootstrap the process. Note, that most graph and kernel-based methods are applied over the space of observations but not over the space of features. One of the reasons is because it is even more difficult to define a distance between features than it is between observations. To circumvent this problem, we propose to learn a feature graph in the latent space of a neural network using feature co-activations as a measure of similarity.

We proceed by creating a graph using feature activation similarity, then applying this graph using Laplacian smoothing for a number of iterations. This converges to a graph of a latent feature space at the level of granularity of the number of dimensions in the corresponding layer.

Our algorithm for learning the graph consists of two phases. First, a pretraining phase where the model is learned with no graph regularization. Second, we alternate between constructing the graph from the similarities of the embedding layer features and further training the network for reconstruction and smoothness on the graph. There are many ways to create a graph from the feature × datapoint activation matrix. We use an adaptive Gaussian kernel,

$$K(z_i, z_j) = \frac{1}{2} exp\bigg(-\frac{||z_i - z_j||_2^2}{\sigma_i^2}\bigg) + \frac{1}{2} exp\bigg(-\frac{||z_i - z_j||_2^2}{\sigma_j^2}\bigg)$$

where σ_i is the adaptive bandwidth for node i which we set as the distance to the k^{th} nearest neighbor of feature. An adaptive bandwidth Gaussian kernel is necessary for general architectures as the scale of the activations is not fixed. Batch normalization can also be used to limit the activation scale.

Since we are smoothing on the graph then constructing a new graph from the smoothed signal the learned graph converges to a steady state where the mean squared error acts as a repulsive force to stop the graph collapsing any further. We present the results of graph learning a biological dataset and show that the learned structure adds interpretability to the activations.

4 Experiments

Through examples, we show that visualizing the activations of data on the regularized layer highlights relationships in the data that are not easily visible without it. We establish this with two examples on fixed graphs, then move to graphs learned from the structure of the data with two examples of hierarchical structure and two with progression structure.

4.1 Fixed Structure

Enforcing fixed graph structure localizes activations for similar datapoints to a region of the graph. Here we show that enforcing a 8×8 grid graph on a layer of a dense MNIST classifier causes receptive fields to form, where each digit occupies a localized group of neurons on the grid. This can, in principle, be applied to any neural network layer to group neurons activating to similar features. Like in FMRI data or a convolutional neural network, we can examine the activation patterns for each localized group of neurons. For a second example, we show the usefulness in encouraging localized structure on a capsulenet architecture [18]. Where we are able to create globally consistent structure for better alignment of features between capsules.

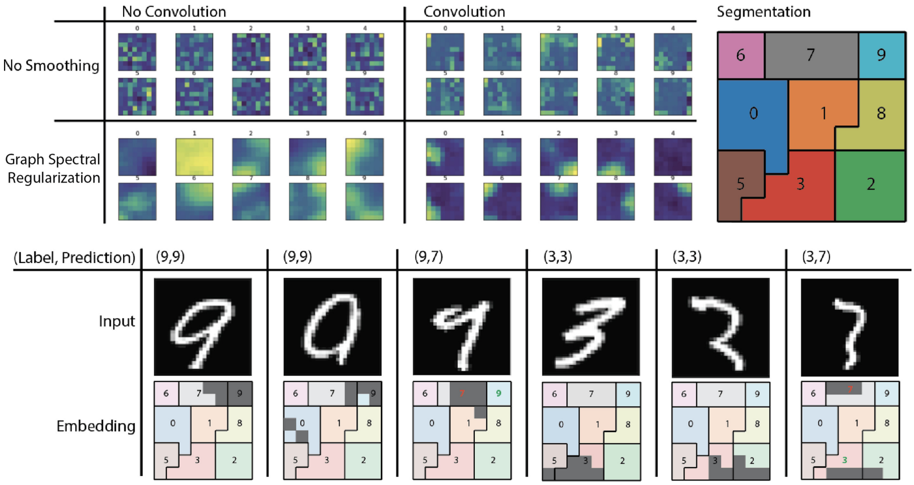

Fig. 1. Shows average activation by digit over an (8×8) 2D grid using graph spectral regularization and convolutions following the regularization layer. Next, we segment the embedding space by class to localize portions of the embedding associated with each class. Notice that the digit 4 here serves as the null case and does not show up in the segmentation. Finally, we show the top 10% activation on the embedding of some sample images. For two digits (9 and 3) we show a normal input, a correctly classified but transitional input, and a misclassified input. The highlighted regions of the embedding space correlate with the semantic description of the input.

Enforcing Grid Structure on Mnist. Without GSR, activations are unstructured and as a result are difficult to interpret, in that it is difficult to visually identify even which class a digit comes from based on the activation pattern (See Fig. 1). With GSR we can organize the activations making this representation more visually distinguishable. Since we can now take this embedding as an image, it is possible to use a standard convolutional architecture in subsequent layers in order to further filter the encodings. When we add 3 layers of 3×3 2D convolutions with 2×2 max pooling we see that representations for each digit are compressed into specific areas of the image. This leads to the formation of receptive fields over the network pertaining to similar datapoints. Using these receptive fields, we can now extract the features responsible for digit classification. For example, features that contribute to the activation of the top right of our grid we can associate with those features that contribute to being the digit 9.

The activation patterns on the embedding layer correspond well to a human perception of the digit type. The 9 that is misclassified as 7 both has significant activation in the 7 region of the embedding layer, and looks visually close to a 7. We can now interpret the embedding layer as a sort of brain map, where the map can map regions of activations, to types of inputs. This is not possible in a standard neural network, where activations are not spatially organized.

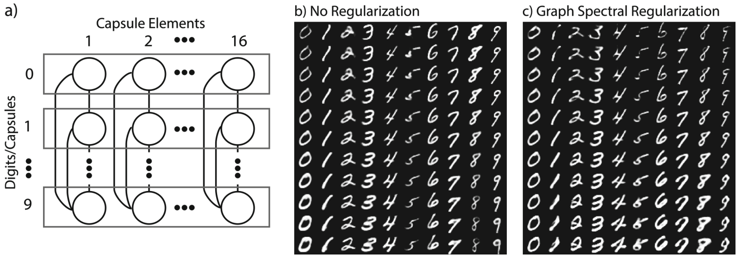

Fig. 2. (a) shows the regularization structure between capsules. (b–c) Show reconstruction when one of the 16 dimensions in the DigitCaps representation is tweaked by $0.05 \in [-0.25, 0.25]$. (b) Without GSR each digit responds differently to perturbation of the same dimension. With GSR (c) a single dimension represents line thickness across all digits.

Enforcing Node Consistency on Capsule Networks. Capsule networks [18] represent the input as a set of vectors where norm denotes activation and each component corresponds to some abstract feature. These elements are generally unordered. Here we use GSR to order these features consistently between digits. We train a capsule net on MNIST with GSR on 16 fully connected graphs between the 10 digit capsules. In the standard capsule network, each capsule orders features randomly based on initialization. However, with GSR we obtain a *consistent feature ordering*, e.g. node 1 corresponds to line thickness across all digits. GSR enforces a more ordered and interpretable encoding where localized regions are similarly organized, and the global line thickness feature is consistently learned between digits. More generally, GSR can be used to order nodes such that features common across capsules appear together. Finally, GSR does not degrade performance much, as can be seen by the digit reconstructions in Fig. 2.

In these examples the goal was to enforce a specified structure on unstructured features, but next we will examine the case where the goal is to learn the structure of the reduced feature space.

4.2 Learning Graph Structure

Using the procedure defined in Sect. 3.1, we can learn a graph structure. We first show that depending on the data, the learned graph exhibits either cluster or trajectory structure. We then show that our framework can learn structures that are hierarchical, i.e. subclusters within clusters or trajectories within clusters. Hierarchies are a difficult structure for other interpretability methods to learn [6]. However, our method naturally captures this by allowing for arbitrary graph structure among neurons in a layer.

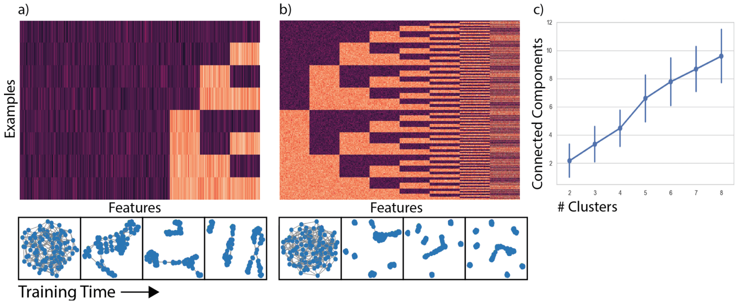

Fig. 3. We show the structure of the training data and snapshots of the learned graph for (a) three modules and (b) eight modules. (c) shows we have the mean and 95% CI of the number of connected components in the trained graph for over 50 trials.

Cluster Structure on Generated Data. We structure our n^{th} dataset to have exactly n feature clusters. We generate the data with n clusters by first creating 2^n data points representing the binary numbers from 0 to $2^n - 1$, then added gaussian noise $N(0, 0.1)$. This creates a dataset with a ground truth number of feature clusters. In the n^{th} dataset the learned graph should have n connected components for n independent features. In Fig. 3 (a–b) we can see how this graph evolves over time for 3 and 8 modules. (c) shows how the learned graph learns the correct number of connected components for each ground truth number of clusters.

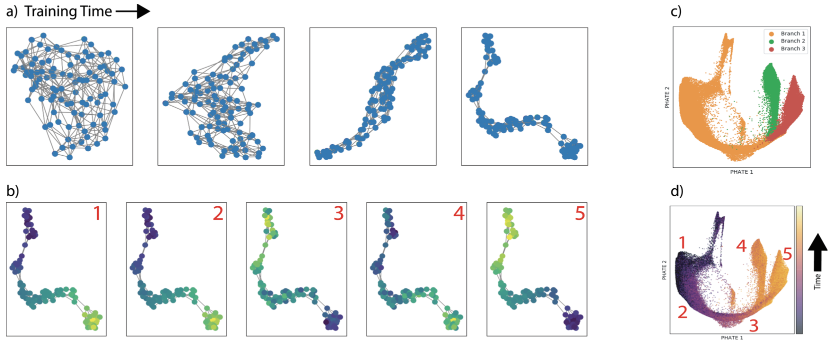

Fig. 4. Shows (a) graph structure over training iterations (b) feature activations of parts of the trajectory. PHATE [15] embedding plots colored by (c) branch number and (b) inferred trajectory location showing the branching structure of the data.

Trajectory Structure on T Cell Development Data. Next, we test graph learning on biological mass cytometry data, which is a high dimensional, single-cell protein dataset, measured on differentiating T cells from the Thymus [20]. The T cells lie along a bifurcating progression where the cells eventually diverge into two lineages (CD4+ and CD8+). Here, the structure of the data is a trajectory (as opposed to a pattern of clusters). We can see in Fig. 4 how the activated nodes in the graph embedding layer correspond to locations along the data trajectory, and importantly, the learned graph is a single connected component. The activated nodes (yellow) move from the bottom of the embedding to the top as T-cells develop into CD8+ cells. The CD4+ lineage is also CD8- and thus looks like a mixture between the CD8+ branch and the naive T cells. The learned graph structure here has captured the transitioning structure of the underlying data.

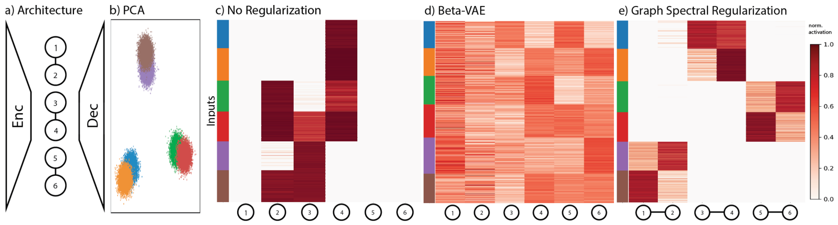

Fig. 5. Graph architecture, PCA plot, activation heatmaps of a standard autoencoder, β-VAE [9] and a graph regularized autoencoder. With relu activations normalized to $[0, 1]$ for comparison. In the model with graph spectral we are able to clearly decipher the hierarchical structure of the data, whereas with the standard autoencoder or the β-VAE the structure of the data is not clear.

Clusters Within Clusters on Generated Data. We demonstrate graph spectral regularization on data that is generated with a structure containing sub-clusters. Our data contains three large-scale structures, each comprising two Gaussian sub clusters generated in 15 dimensions (See Fig. 5). We use this dataset as it has both global and local structure. We demonstrate that our graph spectral regularized model is able to pick up on both the global and local structure of this dataset where disentangling methods such as β-VAE cannot. We use a graph-structure layer with six nodes with three connected node pairs and employ the graph spectral regularization. After training, we find that each node pair acts as a "super node" that detects each large-scale cluster. Within each super node, each of the two nodes encodes one of each of the two Gaussian substructures. Thus, this specific graph topology is able to extract the hierarchical topology of the data.

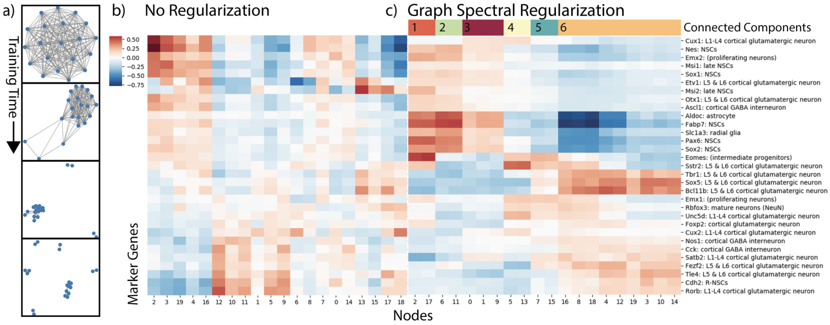

Fig. 6. Shows correlation between a set of marker genes for specific cell types and embedding layer activations. First with the standard autoencoder, then our autoencoder with graph spectral regularization. The left heatmap is biclustered, the right heatmap is grouped by connected components in the learned graph. We can see progression especially in the largest connected component where features on the right of the component correspond to less developed neurons.

Hierarchical Cluster and Trajectory Structure on Developing Mouse Cortex Data. In Fig. 6 we learn a graph on a single-cell RNA-sequencing dataset of over 4000 cells and over 8000 genes. The data contains a set of cells in the process of developing from neural stem cells to full neurons in the mouse brain. While there are many gene modules that contribute to the neuronal development, there are some states that have been studied. We use a list of cell type marker genes to validate our method. We use 1000 PCA components of the data in an autoencoder with a 20-dimensional embedding space. We learn the graph using an adaptive bandwidth gaussian kernel with the bandwidth for each feature set to the Euclidean distance to the nearest neighboring feature.

Our graph learns six components that represent meta features over the gene space. We can identify each with a specific type of cell or related types of cells.

For example, the light green component (cluster 2) represents the very early stage neural stem cells as it is highly correlated with increased Aldoc, Pax6 and Sox2 gene expression. Most interesting to examine is cluster 6, the largest component, which represents development into mature neurons. Within this component we can see a progression from just after intermediate progenitors on the left (showing Eomes expression) to more mature neurons with higher expression of Tbr1 and Sox5. With a standard autoencoder we cannot see progression structure of this dataset. While some of the more global structure is captured, we fail to see the data progression from intermediate progenitors to mature neurons. Learning a graph allows us to create receptive fields e.g. clusters of neurons that correspond to specific structures within the data, in this case cell types. Within these neighborhoods, we can pick up on the substructure within a single cell type, i.e. their developmental trajectory.

4.3 Computational Cost

Our method can be used to increase interpretability without much loss in representation power. At low levels, GSR can be thought of as rearranging the activations so that they become spatially coherent. As with other interpretability methods, GSR is not meant to increase representation power, but create useful representations with low cost in power. Since GSR does not require an information bottleneck such as in β-VAE, a GSR layer can be very wide, while still being interpretable. In comparing loss of representation power, GSR should be compared to other regularization methods, namely L1 and L2 penalties (See Table 1). In all three cases we can see that a higher penalty reduces the model capacity. GSR affects performance in approximately the same way as L1 and L2 regularizations do. To confirm this, we ran a MNIST classifier and measured train and test accuracy with 10 replicates. Graph spectral regularization adds a bit more overhead than elementwise activation penalties. However, the added cost can be seen as containing one matrix vector operation per pass. Empirically, GSR shows similar computational cost as other simple regularizations such as L1 and L2. To compare costs, we used a Keras model with Tensorflow backend [1] on a Nvidia Titan X GPU and a dual Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30 GHz, and with batchsize 256. we observed during training 233 milliseconds (ms) per step with no regularization, 266 ms for GSR, and 265 ms for L2 penalties.

Table 1. MNIST classification training and test accuracies for coefficient selected using cross validation over regularization weights in $[10^{-7}, 10^{-6}, \ldots, 10^{-2}]$ for various regularization methods with standard deviation over 10 replicates.

Regularization	Training accuracy	Test accuracy	Coefficient
None	99.1 ± 0.3	97.5 ± 0.3	N/A
L1	98.9 ± 0.3	97.4 ± 0.4	10^{-4}
L2	98.3 ± 0.3	98.0 ± 0.2	10^{-4}
GSR (ours)	99.3 ± 0.3	98.0 ± 0.3	10^{-3}

5 Conclusion

We have introduced a novel biologically inspired method for regularizing features of the internal layers of dense neural networks to take the shape of a graph. We show that coherent features emerge and can be used to interpret the underlying structure of the dataset. Furthermore, when the intended graph is not known apriori, we have presented a method for learning the graph structure, which learns a graph relevant to the data. This regularization framework takes a step towards more interpretable neural networks, and has applicability for future work seeking to reveal important structure in real-world biological datasets as we have demonstrated here.

Acknowledgements. This research was partially funded by IVADO (l'institut de valorisation des données) [*G. W.*]; Chan-Zuckerberg Initiative grants 182702 & CZF2019-002440 [*S.K.*]; and NIH grants R01GM135929 & R01GM130847 [*G. W.,S.K.*].

References

1. Abadi, M., et al.: TensorFlow: a system for large-scale machine learning. In: OSDI, p. 21 (2016)
2. Achille, A., Soatto, S.: Emergence of invariance and disentanglement in deep representations (2017). arXiv:1706.01350 [cs, stat]
3. Belkin, M., Matveeva, I., Niyogi, P.: Regularization and semi-supervised learning on large graphs. In: Shawe-Taylor, J., Singer, Y. (eds.) COLT 2004. LNCS (LNAI), vol. 3120, pp. 624–638. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27819-1_43
4. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets (2016). arXiv:1606.03657 [cs, stat]
5. Dupont, E.: Learning disentangled joint continuous and discrete representations. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems vol. 31, pp. 710–720. Curran Associates, Inc. (2018)
6. Esmaeili, B., et al.: Structured disentangled representations. In: AISTATS, p. 10 (2019)
7. Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. vol. 2, pp. 729–734. IEEE, Montreal (2005). https://doi.org/10.1109/IJCNN.2005.1555942
8. Higgins, I., et al.: Towards a definition of disentangled representations (2018). arXiv:1812.02230 [cs, stat]
9. Higgins, I., et al.: β-VAE: learning basic visual concepts with a constrained variational framework. In: ICLR, p. 22 (2017)
10. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013). arXiv:1312.6114 [Cs, Stat]
11. LeCun, Y., et al.: Backpropogation applied to handwritten zip code recognition. In: Neural Computation (1989)

12. Liao, R., Schwing, A., Zemel, R.S., Urtasun, R.: Learning deep parsimonious representations. In: NeurIPS (2016)
13. Logothetis, N.K., Pauls, J., Augath, M., Trinath, T., Oeltermann, A.: Neurophysiological investigation of the basis of the fMRI signal. Nature **412**(6843), 150–157 (2001). https://doi.org/10.1038/35084005
14. Min, W., Liu, J., Zhang, S.: Network-regularized sparse logistic regression models for clinical risk prediction and biomarker discovery. IEEE/ACM Trans. Comput. Biol. Bioinf. **15**(3), 944–953 (2018). https://doi.org/10.1109/TCBB.2016.2640303
15. Moon, K.R., et al.: Visualizing transitions and structure for high dimensional data exploration. bioRxiv (2017). https://doi.org/10.1101/120378, https://www.biorxiv.org/content/early/2017/12/01/120378
16. Ogawa, S., Lee, T.M.: Magnetic resonance imaging of blood vessels at high fields: in vivo and in vitro measurements and image simulation. Mag. Reson. Med. **16**(1), 9–18 (1990). https://doi.org/10.1002/mrm.1910160103
17. Ross, A.S., Hughes, M.C., Doshi-Velez, F.: Right for the right reasons: training differentiable models by constraining their explanations (2017). arXiv:1703.03717 [cs, stat]
18. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: 31st Conference on Neural Information Processing Systems (2017)
19. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009). https://doi.org/10.1109/TNN.2008.2005605
20. Setty, M., et al.: Wishbone identifies bifurcating developmental trajectories from single-cell data. Nat. Biotechnol. **34**(6), 637 (2016)
21. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Sign. Process. Mag. **30**(3), 83–98 (2013)
22. Stone, A., Wang, H., Stark, M., Liu, Y., Phoenix, D.S., George, D.: Teaching compositionality to CNNs. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 732–741. IEEE, Honolulu (2017). https://doi.org/10.1109/CVPR.2017.85
23. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks (2019). arXiv:1901.00596 [cs, stat]
24. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks (2013). arXiv:1311.2901 [cs]
25. Zhang, Q., Wu, Y.N., Zhu, S.C.: Interpretable convolutional neural networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8827–8836. IEEE, Salt Lake City (2018). https://doi.org/10.1109/CVPR.2018.00920
26. Zhou, D., Schölkopf, B.: A regularization framework for learning from graph data. In: ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields, vol. 15, pp. 67–78 (2004)