Graph Neural Operators for Learning on Spatial Transcriptomics Data

Junaid Ahmed$^1$ and Alhassan S. Yasin$^2$

$^1$BlueLightAI Inc
$^2$Johns Hopkins University

March 30, 2024
Graph Neural Operators for Learning on Spatial Transcriptomics Data

Junaid Ahmed* and Alhassan S. Yasin†

*BlueLightAI Inc.
E-mail: junaid.ahmed@bluelightai.com
†Johns Hopkins University
E-mail: ayasin1@jhu.edu
Web: https://yasinresearch.com

Abstract—The inception of spatial transcriptomics has allowed improved comprehension of tissue architectures and the disentanglement of complex underlying biological, physiological, and pathological processes through their positional contexts. Recently, these contexts, and by extension the field, have seen much promise and elucidation with the application of graph learning approaches. In particular, neural operators have risen in regards to learning the mapping between infinite-dimensional function spaces. With basic to deep neural network architectures being data-driven, i.e. dependent on quality data for prediction, neural operators provide robustness by offering generalization among different resolutions despite low quality data. Graph neural operators are a variant that utilize graph networks to learn this mapping between function spaces. The aim of this research is to identify robust machine learning architectures that integrate spatial information to predict tissue types. Under this notion, we propose a study to validate the efficacy of applying neural operators towards classification of brain regions in mouse brain tissue samples as a proof of concept towards our purpose and compare it against various state of the art graph neural network approaches. We were able to achieve an F1 score of nearly 72% for the graph neural operator approach which outperformed all baseline and other graph network approaches within the scope of supervised learning.

Index Terms—Article submission, IEEE, IEEEtran, journal, \LaTeX, paper, template, typesetting.

I. INTRODUCTION

A neural network takes in a single input and returns a single output. However, this is not sufficient in capturing systematic relationships by way of the spatial domain. A neural operator [1] takes in a function and outputs a function by way of partial differential equations. These are effective in modeling complex systematic relationships and may be specifically applicable towards interpolation and segmentation tasks. Neural operators seem promising towards spatial transcriptomics not just in offering a broad snapshot of a cell’s tissue-level makeup, but actual spatial positions of those items and their interdependent relationships as well. To obtain a fundamental understanding of neural operators, we consider that the spatial transcriptomics data can be viewed as a continuous function as opposed to a static set of at-time collections. Furthermore, we view the samples of data as being discretized from the continuous function that is defined on the spatial domain \( \mathcal{D} \) of each tissue slide. Neural operators are essentially built on the premise of partial differential equations, and therefore are designed to work with discretized functions.

A. Mathematical Background

The key part of a neural operator is defining a kernel layer that works on a function with learnable parameters. Controlling the properties of the kernel function ultimately controls the layer. There is quite a bit of research to determine optimal kernel functions, for example through the implementation of neural networks and transformers. Additionally, there is some effort being dedicated towards determining a universal approximation kernel function. For the aim of this research, we have identified two kernel functions for the purpose of evaluating the impact of surrounding nodes on the update of the current node: a simple distance kernel involving the Euclidean spatial coordinates, and a more complex graph neural network kernel that captures node associations based on their properties and the graph’s structure.

1) Spatial Kernel: For the spatial kernel based neural operator, a basic distance kernel is utilized, indicated by \( \kappa(d(x, y)) \), where \( d(x, y) \) is a distance measure between nodes \( x \) and \( y \). The kernel function \( \kappa \) measures the distance between nearby nodes and the current node and weighs their update contribution appropriately.

Equation 1 describes the spatial neural operator with a simple kernel.

\[
u(x) = \sigma \left( \sum_{y \in N} \kappa(d(x, y))v(y) \right) \quad (1)
\]

Equation 1 is a variation of the aggregation function used in graph neural networks (GNNs [APPENDIX]) to compute node representations by aggregating information from the node’s local neighborhood. In this particular context, the formula describes how to compute an intermediate representation for a node \( x \), denoted as \( u(x) \), based on the representations of its neighboring nodes \( v(y) \), where \( y \) is a neighboring node of \( x \).

2) Graph Kernel: The graph kernel based neural operator relies heavily on the node attributes and the graph structure. The kernel function \( \kappa_\phi \) captures the relationship between nodes \( x \) and \( y \) and weights their contribution to the update of the current node appropriately. The edge between the node \( x \) and its neighboring node \( y \) is the input for the kernel function,
which produces a weight value. Before being combined with the other contributions from nearby nodes to update the representation vector of the node \(x\), this weight value is multiplied by the representation vector of the neighboring node \(y\), \(v_t(y)\). During the GNN’s training process, the parameters of the kernel function, are discovered.

A graph neural network model may learn this kernel function, enabling it to recognize intricate interactions between nodes and adjust to the unique properties of the graph data. This is significant because, depending on the job at hand, not all nearby nodes are equally significant or instructive. The model may learn which neighbors are most important to the current node and how to weight their contributions using the kernel function. In GNNs, a variety of additional kernel function types may be used, each having unique qualities that might impact the model’s performance. Some kernel functions, for instance, are based on a node-to-node distance metric, while others are based on more complicated functions that may capture non-linear interactions between the nodes. The choice of kernel function can depend on the specific task and the characteristics of the graph being modeled. By enabling the model to learn edge-specific weights that are used to weight the contribution of surrounding nodes in the update of a node’s representation, the kernel function plays a crucial role in the GNN update equation. The model’s performance may be significantly impacted by the kernel function, which is learnt during training.

Equation 2 [2] defines a graph neural operator, or Graph-PDE [3]. For graphs constructed on the spatial domain \(D\), the latter portion of this equation summarized by \(e(x, y)\) is the same as the message passing aggregation of graph neural networks in accordance with the edge features. So simply, the graph kernel, \(\kappa_{\phi}\), is an aggregation of messages. Together, \(\kappa_{\phi}(e(x, y))\) represent the neural network incorporating edge features as input. The additional caveats we consider are \(v_t(y)\) as the node features and \(N(x)\) as the neighborhood of \(x\) according to the graph.

\[
u(x) = \sigma\left(\sum_{y \in N(x)} \frac{1}{|N(x)|} \kappa_{\phi}(e(x, y))v_t(y)\right)
\]

II. RELATED WORKS

Spatial transcriptomics is an emerging field that aims to analyze gene expression patterns in a spatially resolved manner, enabling the identification of complex biological processes that occur at a cellular and tissue level [4]. At the scale this data exists and continues to evolve, this field warrants the development of robust analytical approaches towards current applications of cell-cell interaction, spatial clustering and decomposition, gene imputation, and profiling of localized gene expression, to name a few. The latter application is the focus of this paper as we aim to identify which gene expression distributions of brain regions are conditionally dependent on the spatial location.

There have been many studies to solve the prevalent problem of this profiling space. Machine learning algorithms have shown great promise in approaching solutions in identifying spatially variable genes from different angles [5]. A subset of these machine learning solutions have suggested graph-based methods as notable methods due to their efficacies in aggregating feature information from cells’ locations to improve model accuracy. The foundational graph convolutional network (GCN) developed by Kipf and Welling [6] is an instance of a GNN that has been applied for both unsupervised and supervised learning on spatial transcriptomics data. The GCN is a particular kind of GNN that modifies node representations in accordance with their surrounding nodes in the graph. In spatial transcriptomics analysis, the GCN has been applied to a number of tasks, including cell-cell interaction, cell type marker gene identification, and spatially-aware gene expression prediction [7].

A significant problem persists in that with the gradual emergence of this field, not much labelled data exists to enable supervised learning. As a result, a majority of pre-conducted research in spatial transcriptomics, especially of the graph-enabled variety, is focused on unsupervised or semi-supervised learning tasks like clustering, imputation, and dimensionality reduction techniques [8]. One such approach, named SpaGCN [9], leverages a graph convolutional network (GCN) [APPENDIX] in an unsupervised setting to identify and cluster spatial patterns in genes through the integration of gene expression data, spatial location information, and histology images.

Recent years have seen a rise in the use of graph neural networks (GNNs) for supervised learning tasks using spatial transcriptomics data. GNNs are a type of neural network designed to operate on graphs, which makes them well-suited for analyzing spatial transcriptomics data, where the transcriptome is organized into cells and genes.

One approach proposes the GCNG, or Graph Convolutional Neural networks for Genes [10], [11], which leverages GCNs against genes in a supervised learning task to infer interactions between cells. This approach is promising, and our research further validates that GCNs are exemplary methods towards this data in both performance and accuracy. The drawback to this method is that it is constrained by prior information since it takes known ligand-receptor pairs as labels [12]. Our proposed method is only dependent on strategic normalization and transformation techniques on the available data, and it outperforms the GCN approach in accuracy as well.

For supervised learning on spatial transcriptomics data, GNNs offer tremendous potential. They surpass conventional machine learning techniques and other varieties of neural networks in discovering complicated connections between cells and genes [13]. GNNs, including neural operators, are anticipated to become a more crucial tool for figuring out how cells and tissues are organized spatially as the science of spatial transcriptomics develops.

These studies illustrate the value of taking spatial relationships into consideration when examining gene expression patterns through the lens of graph learning algorithms and highlight the potential of applying graph neural operators for spatial transcriptomics analysis. Graph neural operators could become a potent tool for studying spatial transcriptomics data in learning environments due to their capacity to simulate
connections between cells and genes.

III. MATERIALS AND METHODS

To validate that our materials were prudent for analysis, we established the baseline models we intended to use for evaluating our filtered data. The baseline models included a Logistic Regression (LR) Classifier, a neural network (FCN), a random forest (RF), and an XGBoost (XGB) classifier [Table I]. Each baseline model was presented with the same spatial coordinate information as the GNN models but generally have no inherent way of considering localization through kernel-based approaches for the spatial domain. These were not considered experimental models because, for one, they have been pre-established in industry as best suited techniques in the scope of classification tasks [14], and second, our aim is to validate graph network approaches as viable state of the art classification methods for this type of data. Furthermore, the particular baseline models of Random Forest and XGBoost were chosen due to their robustness in handling imbalanced data, with the latter technique being a gold standard when gauged against tabular datasets [14].

A. Dataset

The goal is to select an optimal spatial transcriptomics dataset. We plan to pursue a supervised classification task to validate the efficacy of using state of the art graph network approaches towards tissue type prediction. For a dataset to be considered towards our intended research, one requirement is for the dataset to have plentiful collected spots across a large feature set of genes. Second, it must have been gathered across a plethora of tissue samples through robust profiling techniques. It is generally difficult to find datasets in this domain which satisfy all of these requirements. Datasets may not be well suited for supervised classification by either the lack of a distinctive target feature(s), or the more prevalent case is that there aren’t enough tissue samples to decipher a decent generalization [15].

We used a central repository of curated spatial transcriptomics datasets [16] in order to find suitable datasets as per our requirements. The “mouse brain atlas”, as is referred by the data curators, [17] we selected for our research contains approximately 34,000 spots and 23,000 genes with 75 samples. This satisfies our requirement of usable datasets by way of number of tissue sections and having been profiled by Illumina spatial transcriptomics profiling techniques. Captured within are the gene expression signatures which define the spatial organization of molecularly discrete subregions. Figure 1 illustrates how the dataset is a “molecular atlas” [17], curated to firstly define the identity of brain regions, as well as establish a molecular code for the mapping and targeting of discrete neuroanatomical domains.

Normalization is a key component to preparing this data before analysis. We required a strategy to distill the large feature set to a trainable set. Additionally, we discovered that introducing sparsity, as per their raw counts set, was not helpful in differentiating variable genes. The curators of the “molecular atlas” [18] provided a pre-normalized set that facilitated our discoveries. As normalization is heavily discussed in literature, please reference [17], [18] for further detail on the extensive normalization procedure performed by the curators of the dataset.

B. Exploratory Data Analysis

The data curators had already undergone a sophisticated filtering and normalization pipeline to map the single cell data onto molecular clusters, and then applied a two-fold dimensionality reduction protocol to arrive at the final gene count. This procedure consisted of an SVM classifier to provide the weights distribution for each molecular cluster, and finally applying independent component analysis to reduce the brain palette [17]. As a result, they were able to determine around 230 of the top genes that would provide the most interesting variation in the dataset.

![Fig. 1. UMAP visualization constructed from the subsampled and normalized gene set \(n \approx 232\) and fit with the spatial information of the gene set across all spots where colors are indicative of the associated brain region. The plot shows strong imbalance in the dataset where a significant number of spots are localized in the Cerebrem as opposed to the other regions. This will be strongly considered towards balanced data preparation for modeling.](image1)

![Fig. 2. Histogram of the total cell count per class.](image2)

Additionally, we discovered that the target feature consisted of 15 classes, yet most of the data was categorized under non-highly differentiable labels, and despite an abundance of classes, there was high class imbalance [Figure 2]. The authors
were using an ontology provided by the Allen Brain Atlas for adult mouse brains [19]. After referencing this ontology, we discovered three high level classes that would optimally separate the data among the 15 classes: the brain stem, the cerebellum, and the cerebrum.

C. Preprocessing

1) Validation set split: As the data we were analysing was sampled based on their associative tissue slides, we couldn’t simply use a randomized sampling approach to split the dataset into its respective modeling sets. We enacted a strategy to use a fraction of the total samples to be held out for validation. With 75 samples in total, we intended to use 7 samples as part of this holdout set. However, there was a clear class imbalance for the target classes. We plotted the samples against their total inherent class representation and annotated those which had representation above a specified threshold of at least 10 classes. Finally, we randomly selected 7 of the highly representative samples to be held out for validation.

2) Filtering and Binning: To round out the preprocessing, we filtered our feature set to only those genes [Figure 3] which the authors for the source dataset identified. Furthermore, we categorized the 15 classes of the target feature into the three isolated classes we discovered through the ABA ontology [19].

3) Reformatting Spots to Graphs: The metadata contains positional information on the classes and spatial coordinates of each coronal section. The normalized data contains the gene expression information of the highly expressive and variable genes. For the data to be processed by the GNN algorithms, this dataset was formatted to be comprised of tuples (G,S,C), where G is a vector of gene expressions (232 derived genes), S=(x,y) are the 2d spatial coordinates, and C is the class (3 classes). As part of the formatting, we applied a radius graph transformation, which creates unweighted edges based on node positions to all points within a given distance, in order to capture and represent points in a grid like manner. The radius parameter was tuned until we found an optimal arrangement of points within each graph.

IV. Results

For our experimental models, we trained a Graph Convolution Network and three neural operator models [APPENDIX]. All models were trained with a balanced class weighting and default hyperparameter values as we considered them to be easily adopted and implemented for these algorithms. Furthermore, the data splits for all models (Section III-C1) and evaluation metrics are maintained constant throughout each of the evaluated models. Both accuracy and F1 metrics were considered for evaluation, however the F1 metric was chosen as the primary evaluation metric as it is the preferred metric in evaluating highly imbalanced sets. Table I reflects the evaluation metrics for each model that we trained against this dataset.

![Dendrogram reflecting heatmap of gene expression values across classes.](image)

![Classification Acitivies and F1 scores for the baseline and GNN models (N=10). The baseline serves to ground and compare this research on common and most widely used approaches in machine learning. This table compares and demonstrates various approaches implemented for the dataset. We see that GraphPDE has the best performance for accuracy and F1-Score for the GNN models, 67.63% and 71.06%. For the baseline approaches we see XGB as 68.45% for accuracy and RF as 62.30% for F1-Score.](image)
we demonstrate through the inclusion of spatial information to these models, we see that better performance can be achieved as in the case of GraphPDE and SpatialGCN.

A. Graph Convolutional Network

The GCN [Figure 4] has a very low performance when factoring in balanced class weights. It predicts all 3 classes to some extent, but generalization of the training is challenging because testing is beyond the discretization grid of the training dataset.

![Predicted vs True class representation of the GCN model.](image)

Fig. 4. Predicted vs True class representation of the GCN model. When we look at the three 3 classes, we see that the GCN gets some predictions right (green). However, due to the variation of resolutions and discretization grid from training and testing sets, it is hard for the network to generalize well. Thus we see that the network gets a relatively lower performance.

B. Neural Operators

Considering all models measured thus far, performance is not a reflection of the power of these networks. They are widely used and very applicable in many domains. However, unlike these networks, neural operators enable learning mappings between function spaces. Also, neural operators are resolution invariant, so the trained operator can be applied on data of any resolution. Traditional neural network models may not generalize well to different unseen discretizations. This is the reason for the improvement in generalization between the neural operators and traditional networks.

For the operator networks, we chose to test both simple and convoluted variations to validate the robustness of each. For our simple approach, we built a kernel network termed SpatialKernel [Equation 2] which incorporated a Gaussian norm and linear layers. We further tested the simple approach by replacing the linear layers with GCN convolution layers for the SpatialGCN model. Finally, for the convoluted approach, we implemented the GraphPDE approach [3] with a graph network as the kernel layer.

1) SpatialKernel: The SpatialKernel model [Figure 5] uses three linear layers, wherein a Gaussian kernel [Equation 2] is computed on the positional features and multiplied to the X tensor before each linear layer. This model performs worse than the GCN, however it seems to be doing a better job in generalizing across the classes as the kernel approach helps extract relevant spatial information.

![Predicted vs True class representation of the SpatialKernel model.](image)

Fig. 5. Predicted vs True class representation of the SpatialKernel model. The model performance is not relatively high but overall the model generalizes better to the different classes. The better generalization behavior due to the kernel approach is reflected in the green colors of correctly labeled classes.

2) SpatialGCN: The SpatialGCN model [Figure 6] combines both the SpatialKernel and GCN such that each layer is a GCNConv layer, and a gaussian kernel is computed before each convolutional layer. This model shows better accuracy than the previous two models, however its predictions are not reflective for all classes.

![Predicted vs True class representation of the SpatialGCN model.](image)

Fig. 6. Predicted vs True class representation of the SpatialGCN model. Generalization for some classes is good, but some classes are not predicted at all.

3) GraphPDE: The GraphPDE network [Figure 7] required some additional customization in the data loading procedures by incorporation of edge attributes. Following this procedural addition, we trained the foundation GraphPDE model against the dataset using only six hidden layers. This model takes in more parameters compared to the other experimental models. This model seems to be the most reliable both in terms
of performance and generalization across classes. It has the highest reported accuracy and F1-score than the other graph network methods. It is slightly better than the XGBoost model (the baseline model to beat) in accuracy and far better than it or any other in F1-score performance.

![Predicted vs Actual: GraphPDE](image)

Fig. 7. Predicted vs True class representation of the GraphPDE model. The overall generalization and performance of this model is better than all other models as this model incorporates edge attribution.

C. Other Methods

Other models for which we considered training against the dataset were supplementary. They were merely used to demonstrate the robustness of the GraphPDE model against high caliber graph neural network models. Discussion on their performance can be found in the APPENDIX section.

V. Conclusion

In this paper, we demonstrated the feasibility of using graph learning techniques to classify mouse brain regions with a particular highlight into the application of neural operators. The GraphPDE model proved to be robust both in performance and generalization across classes. It outperformed all the baselines and similar caliber graph networks with an F1-score of 72%. This research presents graph neural operators as valid performers in supervised classification tasks on spatial transcriptomics data.

A. Discussion and Future Work

The overall study had two goals in mind; (1) analyse the dataset put forth by the “mouse brain atlas” curators [17] and (2) compare the performance of traditional models (baselines) against more state of the art approaches in machine learning to provide enhanced prediction capabilities for Spatial Transcriptomics. First, many of the GNN models of interest employed some semblance of early fusion tactics incorporating multimodal data. As an extension to our work presented in this study, we would consider additional data fusion techniques to compare against neural operators.

Second, the dataset at the point of discovery was one of few nearly perfect curated sets. This dataset has sufficient samples, spots, and genes, however it lacks in generalization. The data comes from one mouse brain. Replicating results gathered to other mouse brains, should they be gathered, may affect the final performance of each model. In a more general sense, we may consider applying this technique to datasets that are not of mouse brains, such as the mouse spinal cord dataset [18], and even cross-species analysis in human anatomy datasets.

Third, the implemented GraphPDE model was shallow with a depth of only six layers. In future works, we would investigate the space of optimal hyperparameters across all models, and further determine whether a deeper GraphPDE network could provide stronger performance. Furthermore, on the note of hyperparameters, there were many models for which we didn’t parameterize certain values as we used the default values provided by the respective algorithms. For example, we could have considered parameterizing the sigma constant in the SpatialKernel model to allow learnability for the most optimal value.

Another method we may consider in the future is late fusion techniques. The methods we utilized reflected early fusion techniques where the data is treated in a multimodal fashion through a single model pass. For future works, we may try to train separate classifiers: one for the feature information and the other either for the positional embeddings or an image classifier for microscope images of the respective samples. Next, we would consider feeding outputs from both of these into a single aggregator model to derive a final output.

Finally, we employed one technique out of many possible neural operator approaches in this research. Other neural operator techniques incorporate state of the art architectures, for example, transformers as a kernel approximation as opposed to the graph network used in the GraphPDE. We may consider using this and other high caliber neural operator approaches for further research.

B. Data Availability

The dataset used and analysed during the current study available from the corresponding author on reasonable request.

ACKNOWLEDGEMENTS

Funding for computational resources was provided by BlueLightAI Inc. The authors thank the staff at BlueLightAI Inc. for their computational support and guidance.

APPENDIX

<table>
<thead>
<tr>
<th>ABBREVIATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABA</td>
</tr>
<tr>
<td>FCN</td>
</tr>
<tr>
<td>GAT</td>
</tr>
<tr>
<td>GIN</td>
</tr>
<tr>
<td>GraphPDE</td>
</tr>
<tr>
<td>GNN</td>
</tr>
<tr>
<td>LR</td>
</tr>
<tr>
<td>RF</td>
</tr>
<tr>
<td>ST</td>
</tr>
<tr>
<td>XGB</td>
</tr>
</tbody>
</table>
Graph Neural Operator Models

Graph Attention Network

The GAT [Figure 8] is not comparable to the top performers in terms of accuracy, nor is it predictive towards all three classes. Furthermore it takes exponentially longer to train, so it is not a worthy tradeoff. As information propagates along the edges of the GAT, the node embeddings for edge-connected nodes within the graph will become closer even though we can’t ensure these nodes have similar labels and features. Thus there is a drawdown to generalization for our dataset.

Fig. 8. Predicted vs True class representation of the GAT model. We see that GAT does not generalize very well and in some cases very poor in predictions of class labels. It is not highly predictive towards all classes.

GraphSAGE

The GraphSAGE [Figure 9] network performs worse than the GAT model and slightly better than the simple SpatialK- kernel model. It is not highly predictive towards all classes. GraphSAGE uses max pooling, and may result in the same output for multi-sets, like for example [a,b] and [a,a,b] will give the same output for max-pooling. Thus, GraphSAGE will fail to identify multi-sets with different structure but the same distinct elements.

Fig. 9. Predicted vs True class representation of the GraphSAGE model. Generalization is not done well for this model and thus poor predictions of class labels is seen.

Graph Isomorphism Network

The GIN network [Figure 10] performs better than the GAT and the GraphSAGE models, which isn’t very telling among the breadth of top GNN model performers, yet is comparable to the suite of actual graph baselines. It is more accurate than the GCN and the loss is minimal compared to other models. It lacks in terms of representation of all three classes, and both with low accuracy and F1-score, it is not a worthy comparator against the GraphPDE.

Fig. 10. Predicted vs True class representation of the GIN model. Generalization is not done well for this model and thus poor predictions of class labels is seen.

REFERENCES


Junaid Ahmed

Junaid has a Master’s degree in the field of Bioinformatics from Johns Hopkins University. He works for a stealth AI startup at the crossroads of topological data analysis, medicine, and generative AI.

Alhassan S. Yasin

Dr. Yasin is a multidisciplinary scientist and a lecturer at Johns Hopkins University. He works at the intersection of Bioinformatics, Artificial Intelligence, Physics, and Engineering. He is also the PI of Yasin Research Group, Web: https://yasinresearch.com.