Supplementary Material for SEPAL: Spatial Gene Expression Prediction from Local Graphs

Gabriel Mejia, Paula Cardenas, Daniela Ruiz, Angela Castillo, Pablo Arbeláez
Center for Research and Formation in Artificial Intelligence
Universidad de los Andes, Bogotá, Colombia
{gm.mejia,p.cardenasg,da.ruiz1,a.castillo13,pa.arbelaez}@uniandes.edu.co

1. Benchmark

	Dataset	
Parameter	**Visium**	**STnet**
`cell_min_counts`	10^3	7×10^2
`cell_max_counts`	10^5	10^5
`gene_min_counts`	10^3	10^3
`gene_max_counts`	10^6	10^6
`min_exp_frac` (ε_{WSI})	0.8	0.01
`min_glob_exp_ frac` (ε_T)	0.8	0.6
`combat_key`	-	Patient

Table 1. Filter hyperparameters for Visium and the STNet dataset

In order to filter our datasets, we define the hyperparameters shown in Table 1. The terms `cell_min_counts` and `cell_max_counts` refer to the minimum and maximum total counts required for a cell to remain in the dataset. Similarly, `gene_min_counts` and `gen_max_counts` specify the minimum and maximum total counts necessary for a gene to be included in the dataset. The parameter `min_exp_frac` determines the fraction of observations in a slide that must express a gene to consider it. Likewise, `min_global_exp_frac` is the fraction of observations across the entire dataset needed to include a gene. Finally, `combat_key` sets the key in dataset to use as batch for ComBat. However, in the case of Visium, this parameter is set to "None" indicating that batch correction is unnecessary.

2. Architecture Optimization

We varied different stages of our model that are enumerated as follows:

1. **Positional encodings**: We decided to sum or concatenate the positional embedding to the patch embedding.

2. **Number of hops** (m): We tuned the number of steps required to reach from one node to another within the graph between 1, 2, and 3 in the Visium dataset, and only 1 in the STNet dataset due to computational cost.

3. **Convolutional graph operator** ($\text{GNN}_i(\cdot)$): We tried different convolutional graph operators from pytorch geometric [2] such as GCNConv[5], SAGEConv[3], GraphConv[6], GATConv[9], GATv2Conv[1], and TransformerConv[8].

4. **Learning rate**: We used 10^{-4}, 10^{-5} and 10^{-6} as learning rate options.

5. **Batch size**: We experimented with batches of size 512, 256, 128 and 64.

6. **Pre-processing stage**: We tried to progressively reduce the dimensionality of our graph input by adding an MLP with the following options of hidden channels h_i:

   - No MLP
   - $h_i = \{d_{\text{emb}}, 512\}$
   - $h_i = \{d_{\text{emb}}, 512, 256\}$
   - $h_i = \{d_{\text{emb}}, 512, 256, 128\}$

   Where d_{emb} refers to the embedding dimension (768 if the positional encoding is sumed or 1536 if it is concatenated).

7. **Post-processing stage**: We tried to progressively decompress the graph network output adding an MLP with the following options of hidden channels h_o:

   - No MLP
   - $h_o = \{128, n_g\}$
   - $h_o = \{64, 128, n_g\}$

   Where n_g is the number of genes to predict, set at 256.

Method	ResNet18	DenseNet-121	ConvNeXt-T	WideResNet-50	MobileNetV3-S	ResNeXt-50	ShuffleNetV2_X0_5	Swin-T	ViT-B-16
MAE	0.653	0.641	0.653	0.649	0.645	0.642	**0.635**	0.650	0.638
MSE	0.761	0.745	0.754	0.756	0.736	0.739	**0.725**	0.751	**0.725**
PCC-Gene	0.296	**0.379**	0.294	0.323	0.338	0.341	0.354	0.303	0.347
R2-Gene	0.051	0.071	0.050	0.060	0.084	0.080	**0.094**	0.055	0.086
PCC-Patch	0.923	0.925	0.924	0.924	0.925	0.925	**0.927**	0.924	**0.927**
R2-Patch	0.843	0.846	0.844	0.844	0.846	0.846	**0.850**	0.844	0.849
Params (M)	11.7	8.0	28.6	68.9	2.5	25.0	1.4	28.3	86.6
Batch Size	64	320	128	320	128	64	256	256	320
lr	1×10^{-2}	1×10^{-3}	1×10^{-4}	1×10^{-2}	1×10^{-2}	1×10^{-3}	1×10^{-2}	1×10^{-5}	1×10^{-4}

Table 2. Results of the extensive experimentation performed on image encoder selection over the Visium dataset. The best performance is written in **bold**, and the second best result is underlined for each metric.

MLP pre	GNN	MLP pos
-	d_{emb}, n_g	-
-	d_{emb}, 512, n_g	-
-	d_{emb}, 512, 256, 128	128, n_g
-	d_{emb}, 512, 256, 128, 64	64, 128, n_g
d_{emb}, 512	512, 256, n_g	-
d_{emb}, 512, 256	256, 128, n_g	-
d_{emb}, 512, 256	256, 128, 64	64, 128, n_g
d_{emb}, 512	512, 256, 128	128, n_g
d_{emb}, 512, 256, 128	128, 128, 128, 128	128, n_g

Table 3. Hidden channels dimensions for preprocessing MLP, Graph Neural Network and postprocessing MLP.

8. **Graph hidden channels**: To fit the dimensions of pre and post-processing stages, we follow an autoencoder-like architecture by powers of 2. The options of hidden channel dimensions go from 1 to 4 graph convolutional layers as shown in Table 3.

With this systematic procedure, we generate 3888 hyperparameter combinations from 324 module variations in the Visium dataset, and 1296 hyperparameter combinations from 108 module variations in the STNet dataset.

3. Image Encoder Selection

To establish the architecture of our image encoder we experiment with nine of the most recent and popular image classification models, replacing their respective last layer to predict the expression of the n_g genes of interest. With each potential image encoder we do a grid search modifying the learning rate inside $[10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$ and the batch size inside $[320, 256, 128, 64]$ to achieve the best possible performance in the Visium dataset. Table 2 shows the best results for each one of the architectures. The results show that ShuffleNet and ViT have similar performance in most metrics, achieving the best results among almost all the architectures examined. However, ViT has a slightly better MSE (0.7245 Vs 0.7249), which is the selection criteria. Hence, we build SEPAL applying ViT as the image encoder $I(\cdot)$.

Method	STNet	EGN	EGGN	HisToGene	SEPAL
MAE	0.731	0.785	**0.651**	0.742	0.663
MSE	0.955	1.077	**0.745**	0.967	0.789
PCC-Gene	0.106	-0.028	**0.297**	-0.048	0.292
R2-Gene	-0.189	-0.304	**0.059**	-0.191	0.012
PCC-Patch	0.911	0.901	**0.926**	0.901	0.922
R2-Patch	0.796	0.770	**0.845**	0.800	0.836

Table 4. Results of SEPAL along with state-of-the-art models when trained in the noisy version of the Visium dataset. The best performance is written in **bold**, and the second best result is underlined for each metric.

4. Training with Noisy Data

To validate our denoising approach, we replicate the experimentation pipeline (training protocol optimization) for the main results in the Visium dataset. However, this time the training data does not pass through the modified adaptive median filter resulting in a noisy training dataset. The results can be observed in Table 4 and, outstandingly, even with just 5.3% of noisy data, the performance of all methods degrades significantly. This observation validates our missing data imputation protocol disregarding the computational method and sets a best practice for future works. Importantly SEPAL falls to the second position of the ranking because the noisy data greatly affects the computation of $\bar{y}_{\text{train}}$ producing a systematic bias in the predictions.

References

[1] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021. 1

[2] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. 1

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 1

[4] W Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, 2007. 4

[5] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1

[6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1

[7] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 4

[8] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020. 1

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 1

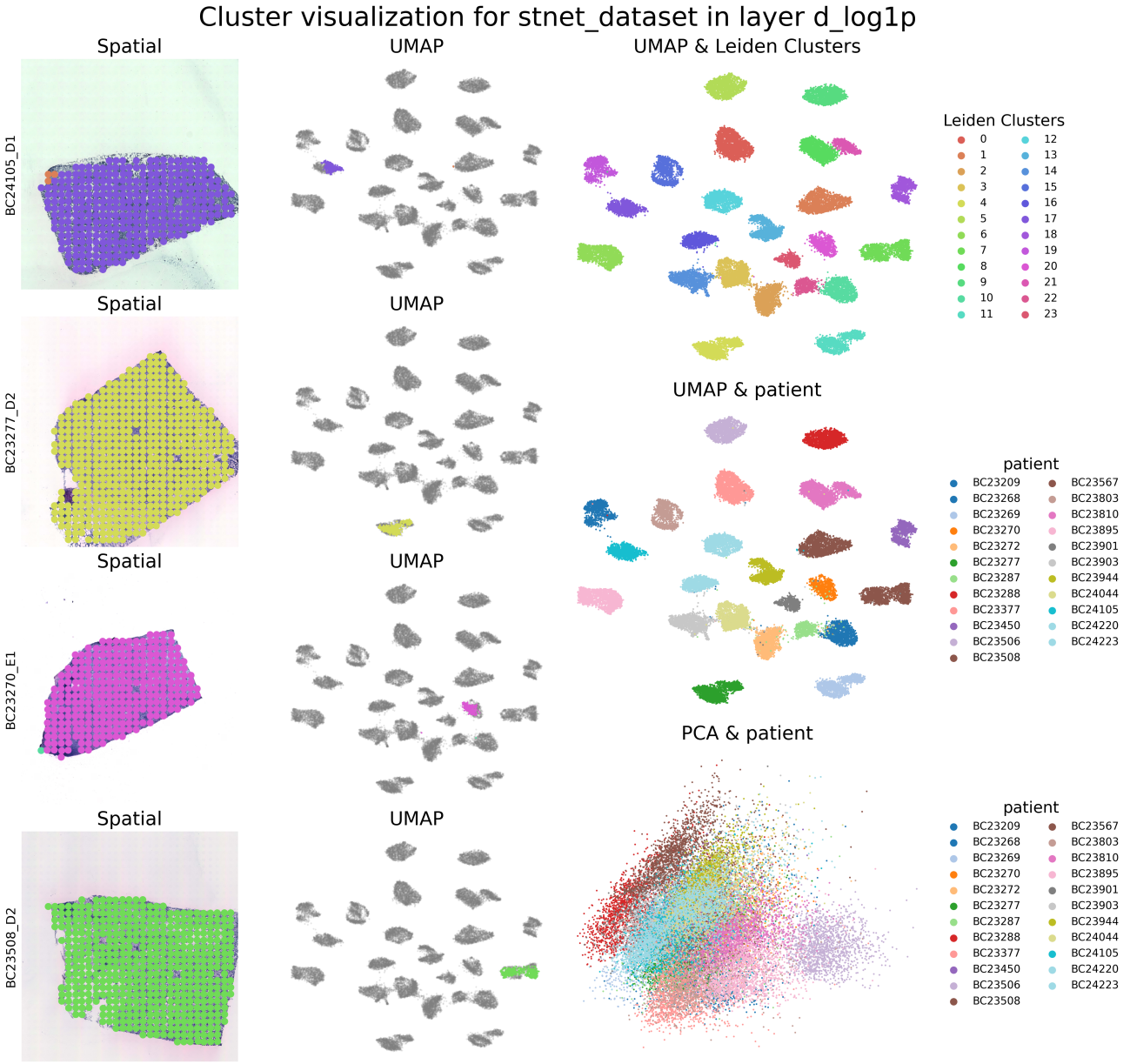

Figure 1. UMAP dimensionality [7] reduction and clusters visualization for STNet dataset without batch correction. As strong batch effects are seen depending on the patient, ComBat [4] was applied with the patient as the batch key in the processing pipeline of this dataset.

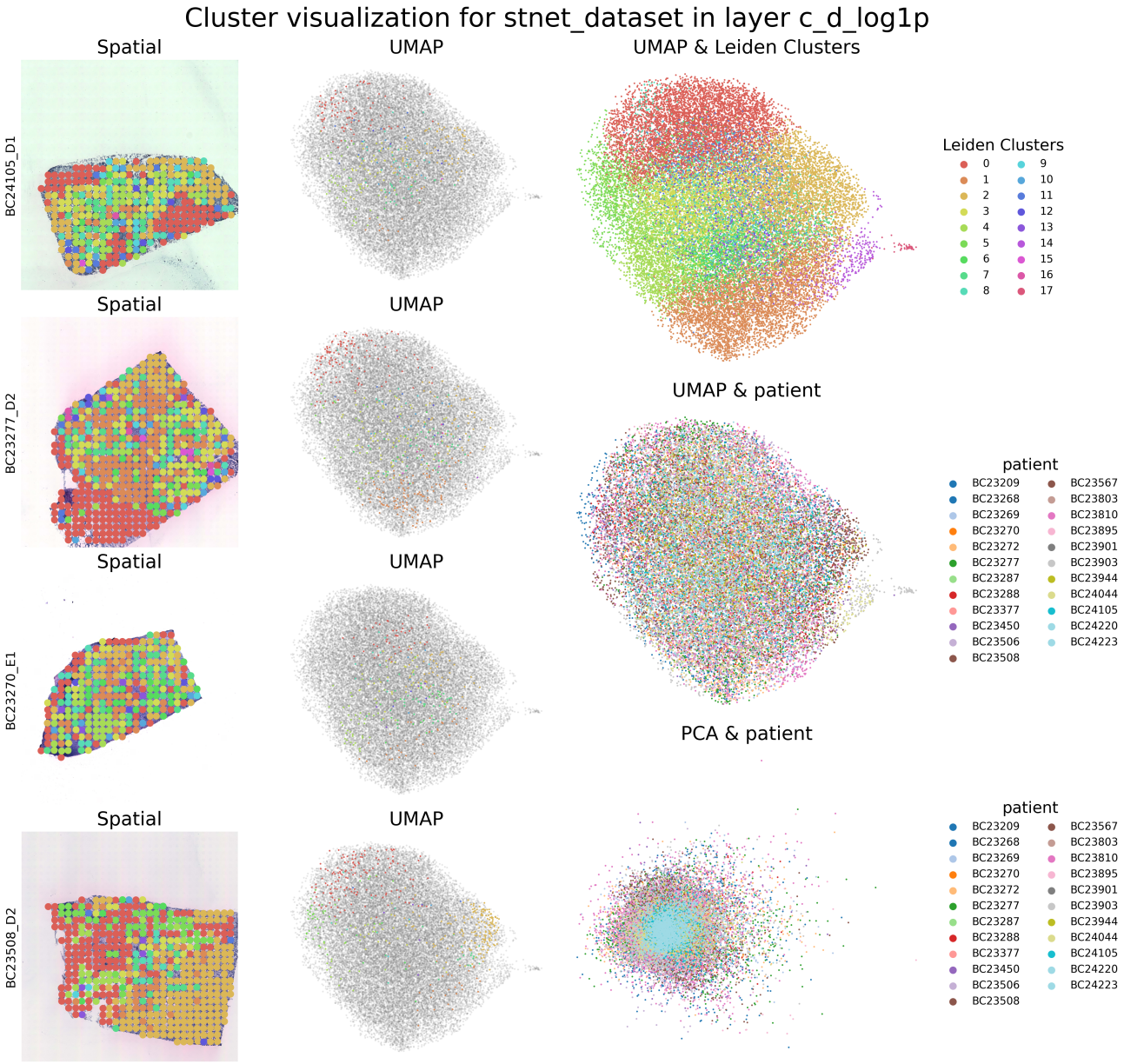

Figure 2. UMAP dimensionality reduction and clusters visualization for STNet dataset after ComBat batch correction.

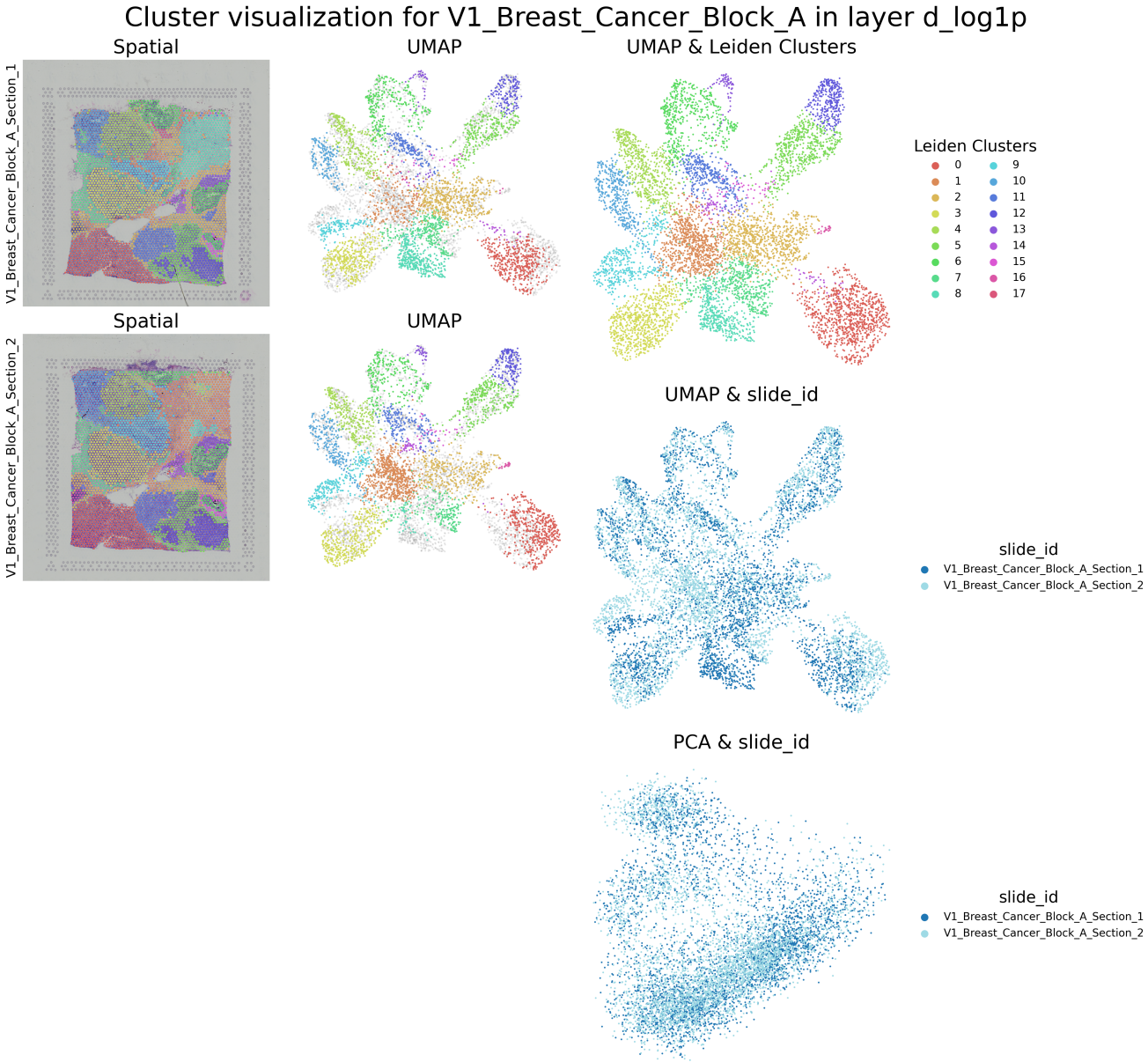

Figure 3. UMAP dimensionality reduction and clusters visualization for Visium dataset without batch correction. As no evident batch effects are present, ComBat was not applied in the processing pipeline of this dataset.

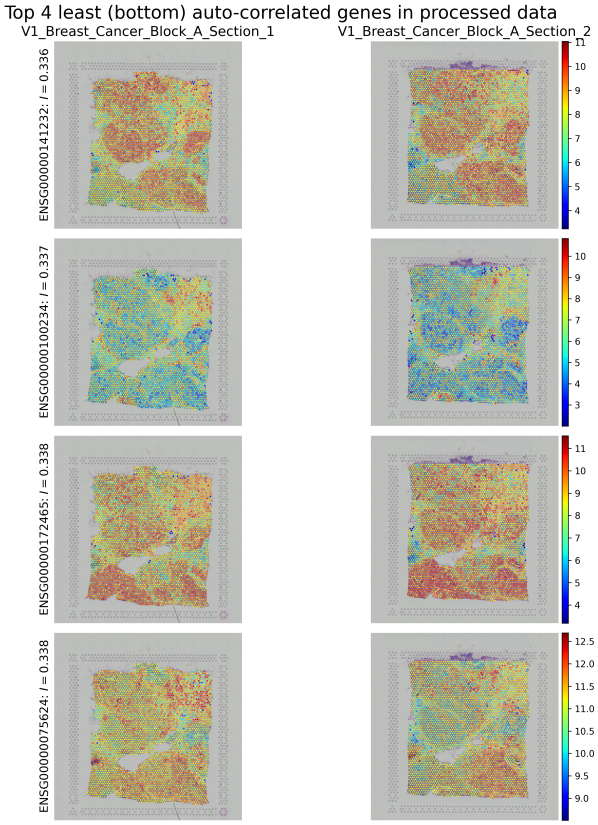

Figure 4. Visualization of the four least auto-correlated genes in processed data for Visium in both whole slide images.

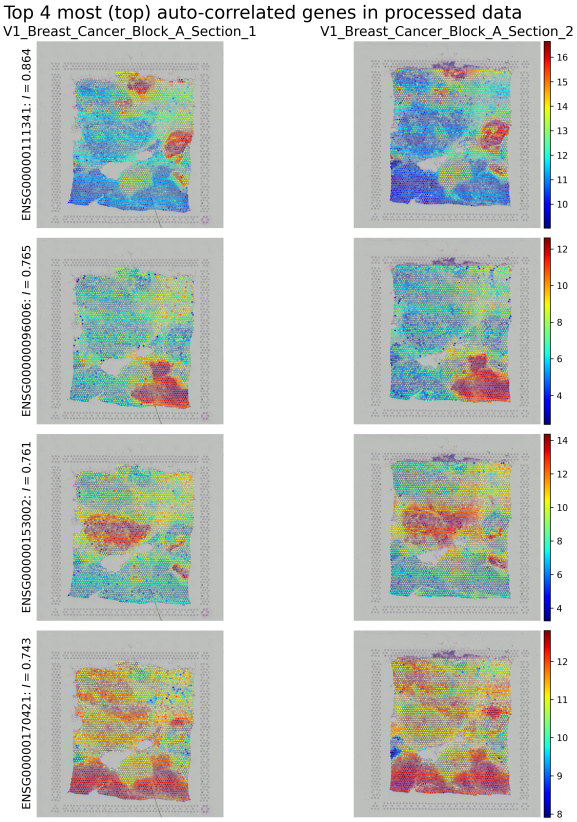

Figure 6. Visualization of the four most auto-correlated genes in processed data for Visium in both whole slide images.

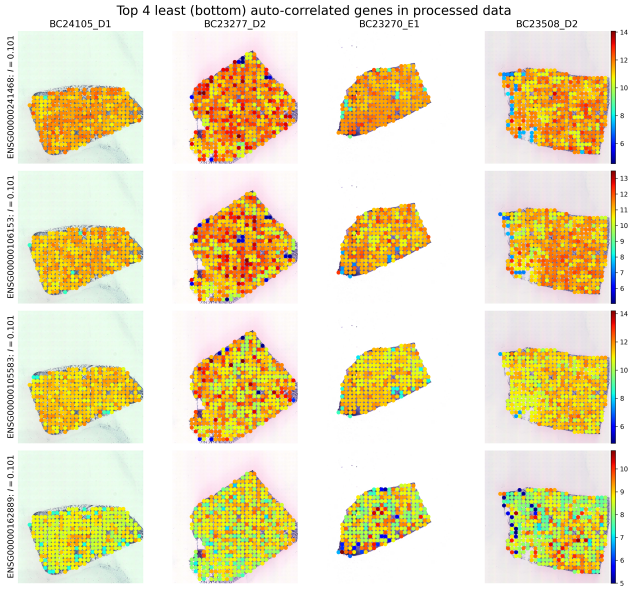

Figure 5. Visualization of the four least auto-correlated genes in processed data for STNet in four whole slide images.

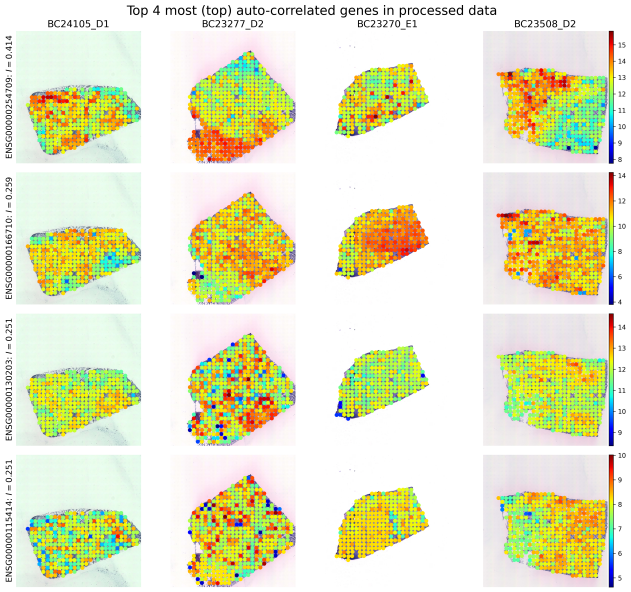

Figure 7. Visualization of the four most auto-correlated genes in processed data for STNet in four whole slide images.