Supplementary Methods:

In Silico Pattern Generation:

In silico patterns were generated using 7 different pattern generators: Undifferentiated, Differentiated, Inside-Out, Outside-In, Random, Globular, and Snaked. The differentiated patterns were created by assigning a differentiated state to all except 5% of the cells. The undifferentiated states were created by assigning an undifferentiated state to all but 5% of the cells. The 5 % margin was incorporated to allow for some of the error resulting from the network digitization algorithm. Outside-In patterns were generated using a radius parameter; all cells within the radius were set to undifferentiated while all cells outside were set to differentiated. Inside-Out patterns were generated using a radius parameter; all cells inside the radius were set to differentiated, while all cells outside were set to undifferentiated. Random patterns were generated by randomly setting a fraction of the cells to a differentiated state. Globular patterns were created using two parameters: a seed number, and expansion number. The seed number governed how many differentiated cell clusters were found initially, and the expansion parameter governed how many layers of nearest neighbors were converted into a differentiated state. For example, an expansion parameter of 2 would turn all cells within 2 network connections away to a differentiated state. Snaked parameters were generated using two parameters: the number of seeds, and the elongation of each seed. The number of seeds again denoted the number of initial starting locations that were differentiated. Each of these conditions was then extended to a length of n, by picking a random neighbor and converting its state to differentiated and repeating the process until the target length was reached.

PCA:

Principal component analysis was performed using *sklearn* for the python programming language and used as a dimensional reduction technique for data visualization. The python package *Matplotlib* was used to plot all PCA plots, while the heatmaps displaying component information where created with custom written code using a combination of the python packages *numpy* and the *PIL*. All data points were mean centered and unit variance scaled as required by the PCA algorithm using the scale function from *sklearn*. The PCA algorithm relied on singular value decomposition, leading to multiple fitted estimators displaying the data with principal components reversed. Automatic dimension fitting was performed using the method of Thomas P. Minkas as needed(44).

Network Reconstruction via Cell Profiler:

Cell Profiler (http://www.cellprofiler.org/)(45) was used to analyze all of the 2D samples. For confocal analysis, images were imported, split into their component channels (i.e. red, blue, green) and cell nuclei were detected using a local Otsu thresholding approach to provide a binary mask. Clumped nuclei were resolved using the “intensity” module, followed with the “propagation function” within Cell Profiler, which led to extended objects that were termed “cells”. The green signal of each object (indicative of Oct4 expression) was measured and reported as the median and mean value. Additionally, the number of adjacent nearest neighbors was measured and the data was then exported to a python script that reconstructed the networks by using a KDTree implementation from scipy. Network images were generated using the python imaging library (PIL). Annotation was performed by thresholding the Oct4 intensity values. In the case of multiple EBs per image, the networks were split such that each individual network contained only one EB.

Network Reconstruction in 3D:

First confocal images were read by ImageJ, merged into a single RGB image and then saved as an image sequence. The image sequence was read using python and converted into a memory mapped array via the numpy package, which allows for analysis of large arrays that would normally exceed the amount of memory present in the computer's RAM. Images were split into respective red, green, and blue components and then denoised using a Gaussian filter from scipy's ndimage package. Initial thresholding was performed on the blue channel using a global Otsu approach from the python package skimage to identify grouped nuclei. Local maxima detection was performed to segment nuclei using the python package skimage and once detected, a merging step was performed to identify local maxima that were too close to each other, and these were merged into a single new local maxima point. The local maxima points were converted into seeds for nuclei detection and served as the subsequent nodes in the network. Connections were formed using a nearest neighbor approach using a KD Tree implementation from scipy, in which only neighbors within a certain distance (twice the cell radius) were connected to each other. The cells were then annotated by computing the average red and green values within radii around the points. A global threshold over all images was established for the red and green channels using an Otsu thresholding approach over all of the nodes. The networks were filtered to remove unconnected subnetworks, and further filtered using a nuclei quality (based on intensity and size of the nuclei) metric to produce the final annotated network.

Classification:

The following classification methods from the python package *sklearn* were used: SGD, NuSVC, SVC, linear SVC, Decision Tree, K Nearest Neighbors. In all cases, classifiers were trained using a test data set, followed by subsequent classification of a test set for metric evaluation. The data set was split into a test and training set using a K-Fold splitting strategy. Grid searches for the sets were also performed as outlined in the supplementary methods. This code set up the grid searches which were then subsequently run and returned the best trained classifier. These classifiers where then evaluated using the following criterion: recall, precision, f1_score, area under the curve, average precision, accuracy. Accuracy was computed as the fraction of completely accurate predictions. The average precision scores and the area under the curve were derived from the precision-recall curve, where the average precision is the average value over the curve, and the area was the integral of the entire curve. The precision score measured the ability of the classifier to not label a negative sample as positive. The recall score was the ability of the classifier to find all positive samples. The f-measure was a weighted harmonic mean between the precision and recall scores and was computed as F_1 = 2* (precision*recall) / (precision + recall).

Feature Elimination (variable trimming):

Variables were eliminated from the data set by examining the variance present within each variable type. This was accomplished by examining the variable distributions on a boxplot (where each data point was an individual sample) and the clustering of variables shown by hierarchical clustering. This allowed variables which did not vary over samples to be identified and eliminated via the box plot test, and also to identify groups of variables which introduce confounding trends into the data set. Typically the box plot test was used to remove variables which did not change between samples (displaying a mean with an extremely tight standard deviation). In the case of the hierarchical clustering, if groups of variables segregated into larger groups and were separated by a normalized distance of >0.7 this test was used to determine which grouping of variables should be removed (typically a smaller isolated cluster of variables).

Quantitative Comparisons between Multivariate Trajectories:

Principal component analysis was performed to identify the most significant components of variation in the data. The data were then only compared along these axes, as utilizing the PC weights could account for the importance of individual measurements in comparing distances. Furthermore, a clustering algorithm (k-means, unsupervised) was performed on each set of data to determine if the subset of data trying to be fitted (typically a time point) could be split into smaller clusters of data to take care of multimodal data. This is an analogous method to creating a GMM (Gaussian mixture model) for each data subset. Individual peaks of the Gaussian were compared in a pointwise manner to determine the distance between two sets of data, resulting in a set of distance measurements which could be collected form a given set of pairwise data sets. Typically this was performed by aligning two data sets in time. Binning by time can be excluded to compare how closely one set of data matches another at each point in the series and was denoted as the absolute distance measurement.

Modeling Local Interactions:

Computational modeling was carried out using a previously established framework with some slight modifications. A KDTree implementation as provided in the scipy.spatial package for python was used for detecting and resolving collisions. The form of the rule functions was changed from previous work into a more classical activation and deactivation function:

$P(x) = \alpha$ Equation 1: Random Feedback

$P(x) = \frac{k * norm\_d^{n\_p}}{k1^{n\_p} * norm\_d^{n\_p}}, norm_d = \frac{d}{\epsilon}$ Equation 2: Positive Feedback

$P(x) = \frac{k2}{k3 + \left(\frac{norm\_u}{k4}\right)^{n\_n}}, norm_u = \frac{u}{\varepsilon}$ Equation 3: Negative Feedforward

$P(x) = \frac{k}{k2 + \left(\frac{norm\_u}{k3}\right)^{n\_n}}$ or $\frac{k * norm\_d^{n\_p}}{k2^{n\_p} * norm\_d^{n\_p}}$ Equation 4: Combined Feedback

Where α denotes basal random differentiation, ε denotes the number of neighbors, k1 and k denote the response levels of the sigmoidal functions, and norm_u and norm_d represent the normalized number of neighbors in the undifferentiated and differentiated states respectively. These values were simulated over a variety of parameter ranges (Supplementary Table 1). These rules were considered to be satisfied if a randomly generated value was less than the calculated probability.

Modelling Diffusion:

Modelling of diffusive species was carried out using a simple Euler forward integration scheme for solution. Though faster solutions exist, the forward integration scheme is stable even when solving reaction diffusion equations with spatially heterogeneous consumption and production terms. The diffusion equation was solved using three separate convolution kernels for each independent direction: x, y and z. The solution was simulated according to Equation 5.

$$C(t) = \sum_0^t C_{t-1} + D * dt * (lx + ly + lz) + p - q \qquad \text{Equation 5}$$

Where D is the diffusion coefficient, dt the time of integration, p the source term, and q the consumption term, and l_x, l_y, l_z are the solutions to the 1-D diffusion kernel in x, y, and z respectively. In the case of the source and sink terms, both are assumed to be independent of the concentration, and both are calculated by summing the total contributions of all cells at a given location on the grid. The grid size (15 μm) was chosen so that at most 2-3 cells could share the space at the same time. The summation indicates that the solution is solved iteratively until either the upper time limit is met, or a steady-state convergence is reached. Steady-state was defined as no appreciable change (1E-5 of the concentration value) in the concentration gradient from a given time step to the next. All coefficients and constants were defined such that the final units of the concentration were in μM, with a spatial resolution in μm. The time step dt, was subjected to the constraint in Equation 6 for conditional stability of the integration scheme:

$$dt = \frac{.5}{D(\frac{1}{\Delta x} + \frac{1}{\Delta y} + \frac{D1}{\Delta z})} \qquad \text{Equation 6}$$

Where x, y, z represent the spatial resolution of the grid on which the solution was solved. All convolution was carried out using scipy.ndimage convolve function, while a specific gradient class was created for handling diffusion of different soluble species. The rules for differentiation based on diffusion were implemented using classic inhibition equations. In the case of positive induction, where the soluble factor induced differentiation, the probability was defined as is shown in Equation 7. In the case of inhibition, where the soluble factor inhibited differentiation, Equation 8 was used. In the case of competing differentiation, both rules were applied, however the second soluble factor was only secreted by differentiated cell types.

$$P(x) = \frac{1.0}{k1^n + v1^n} \qquad \text{Equation 7}$$

$$P(x) = 1 - \frac{1.0}{k2^n + v2^n} \qquad \text{Equation 8}$$

In this context, v1 and v2 represent the values of the two concentrations at the point in space occupied by the cell, k1 and k2 represent the set points at which the cells begin to respond, and coefficient n denotes the width of the response distribution. Additionally, a counter was implemented to keep track of the time delays associated with each soluble signaling factor, thereby essentially representing a signal duration parameter that helped to modulate the response time of cell to given soluble signals.

Parameter ranges:

A range of parameters for the various different rules was investigated to provide a coherent sampling of the relevant parameter space. These parameter ranges are summarized for each rule in Supplementary Table 1. For the paracrine rules, evaluation of the consumption to production coefficients ratio (denoted p/q) was performed separately as a test before large scale simulation were run. Taken together all of the parameter sets took ~ 2500 hours of cumulative simulation time to run, which on a 64 core cluster was equivalent to approximately 380 hours of run time (~ 15 days) and generated roughly 15 terabytes of data.

Cell Culture:

A Murine embryonic stem cells (D3) were expanded on 100-mm tissue culture plates coated with 0.67% gelatin in Dulbecco’s modified Eagle’s medium (DMEM) supplemented with 15% fetal bovine serum(FBS) (Hyclone, Logan, UT), 2mM L-glutamine (Mediatech), 100 U/ml penicillin, 100 ug/ml streptomyocin, and 0.25 ug/ml amphotericin (Mediatech), 1x MEM nonessential amino acid solution (Mediatech), 0.1 mM 2-mercaptoethanol (FisherChecmical, Fairlawn, NJ), and 10^3 U/ml leukemia inhibitory factor (LIF) (Chemicon International, Temecula, CA). Cells were passaged every 2-3 days prior to reaching 70% confluence.

EB Formation and Culture:

To initiate ESC differentiation, a single cell solution was obtained via dissociation in a 0.05% trypsin / 0.53 mM EDTA solution. Embryoid bodies (EBs) were formed via forced aggregation of single cells into 400 µm diameter PDMS microwells (AggreWell), with approximately 1000 cells per well. After 20 hours of formation, EBs were removed from the microwells and maintained in suspension on a rotary orbital shaker at 65 rpm, with approximately 1500 EBs per plate(26). EBs were fed via gravity-induced sedimentation and 90% of the media was exchanged every other for up to 7 days of differentiation. EBs were formed and cultured in the standard growth media without LIF.

Immunostaining and Confocal Microscopy:

EBs were fixed in 10% formalin for 45 minutes, permeabilized for 30 minutes in 1.0% TritonX-100, re-fixed in formalin for 15 minutes, and blocked in blocking buffer (2% bovine serum albumin, 0.1% Tween-20 in PBS) for 3 hours. Samples were stained with a goat Oct4-antibody (Santa Cruz, 1:100 dilution) overnight at 4 °C. After three washes in blocking buffer, EBs were subsequently stained with a secondary donkey anti-goat Alexa Fluor 488 conjugated antibody (1:200 Santa Cruz) for 4 hours. Staining with Alexa Flour 546 Phalloidin (1:20 Molecular Probes) and Hoescht (1:100) was performed concurrently for 25 minutes. Samples were washed, resuspended in blocking buffer, and imaged using a Zeiss LSM 510 NLO Confocal Microscope. A single image was taken at the top of each EB and at a depth of 30 µm into the EB; for each timepoint, a minimum of 25 images were obtained.

Cichlid maintenance and culture:

Several species of East African Cichlids were kept as brooding populations in 40 gallon tanks, with two species per tank configurations, for a total of 30 to 40 individual fish. The male to female ratio was typically 1:4 up to 1:10, depending on species. Fishes were allowed to spawn naturally, then broods were taken from the female's mouth approximately 24 hours post fertilization (hpf). A brood consisted of 20 to 80 eggs, depending on the species. Broods were grown in 150 mL flasks, in a mixture of tank water from which the mother lives, and methylene blue, to prevent fungal growth. A subset of individuals was taken from each brood at 36 hpf and at 4 hour intervals until 48 hpf (denoted, 0, 4, and 8 hours in the text) to cover the entire duration of gastrulation. Embryos were fixed in 4% paraformaldehyde (PFA) at each time point of interest.

Cichlid Embryo Staining and Confocal Imaging:

After fixation, expression of *dlx3b* was visualized using whole mount *in situ* hybridization, using a modification of previously published methods(46). The gene was visualized using Fast Red (naphthol chromogen, Roche Diagnostics). After *in situ* hybridization, embryos were immunostained for pSmad 1,5,8 protein, using previously published protocols(47). Embryos were then bathed in Vectashield (Vector Labs) containing DAPI and placed in a specially built mold to hold embryos upright. Embryos were scanned using a Zeiss LSM 700-405 confocal microscope and images were processed using LSM 700 software and Image J.

Path Finding Algorithms:

Path finding algorithms were used to assess the likely flow of information through a system. Briefly, a set of waypoint nodes were chosen for the algorithms to pass through in a given order. For cichlid fish analysis, 3 waypoint nodes where chosen, a start node, an end node, and one in the middle of the gastrulation process. Simulations were run to find likely paths or flows of information by assigning a probability function to designate which node should be chosen next. In this case the PDF was directly related to the distance in the PCA reduced metric space. Thus, the resultant paths were reduced to the shortest distance representations in metric space constrained by passing through all waypoint nodes.

Grid search parameters:

NuSVC:

```
nus = [1E-7, 1E-6, 1E-5, 1E-4, 1E-3, 1E-2, .1],
params = [{'nu':nus, 'kernel': ['linear']},
          {'nu':nus, 'gamma':np.logspace(-5,0,num=6), 'kernel':['rbf']},
          {'nu':nus, 'gamma':np.logspace(-5,0,num=6), 'degree':np.arange(2,5), 'kernel':['poly']},
          {'nu':nus, 'gamma':np.logspace(-5,0,num=6), 'kernel':['sigmoid']}]
 classifier = NuSVC(probability = True)
 gs = GridSearchCV(classifier, params)
```

SVC:

```
params = [{'C':np.logspace(-5,5,num=11), 'kernel': ['linear']},
          {'C':np.logspace(-5,5,num=11), 'gamma':np.logspace(-5,0,num=6), 'kernel':['rbf']},
          {'C':np.logspace(-5,5,num=11), 'gamma':np.logspace(-5,0,num=6), 'degree':np.arange(2,5), 'kernel':['poly']},
         {'C':np.logspace(-5,5,num=11), 'gamma':np.logspace(-5,0,num=6), 'kernel':['sigmoid']}]
classifier = SVC(probability = True)
gs = GridSearchCV(classifier, params)
```

SGD:

```
params = [{'alpha':np.logspace(-5,5,num=11), 'loss':['log']},
          {'alpha':np.logspace(-5,5,num=11), 'loss':['modified_huber']},
          {'alpha':np.logspace(-5,5,num=11), 'loss':['perceptron']},
          {'alpha':np.logspace(-5,5,num=11), 'loss':['squared_hinge']}]
classifier = SGDClassifier(shuffle = True)
 gs = GridSearchCV(classifier, params)
```

linear SVC:

```
params = [{'C':np.logspace(-5,5,num=22)}]
classifier = LinearSVC()
gs = GridSearchCV(classifier, params)
```

K Nearest Neighbors:

```
params = [{'n_neighbors':np.arange(1,20)}]
classifier = KNeighborsClassifier()
gs = GridSearchCV(classifier, params)
```

Decision Tree:

```
params = [{'criterion':['entropy'], 'max_features':np.arange(1, len(ml))},
              {'criterion':['gini'], 'max_features':np.arange(1, len(ml))}]
 classifier = DecisionTreeClassifier()
 gs = GridSearchCV(classifier, params)
```

Metric Definitions:

Binary Metrics (Oct4+, Oct4-):

Oct4+_clust_# - the number of Oct4 + clusters (a cluster is defined as more than a single node)
Oct4+_size_avg – the average radius of the Oct4+ clusters
Oct4+_size_std – the standard deviation in the radius fo the Oct4+ clusters
Oct4+_nd_cnt_avg – the average number of nodes of the Oct4+ clusters
Oct4+_nd_cnt_std – the standard deviation in the number of nodes of the Oct4+ clusters
Oct4+_rad_dist_avg – the average radial distance of the Oct4+ clusters
Oct4+_rad_dist_std – the standard deviation of the radial distances of the Oct4+ clusters
Oct4-_clust_# - the number of Oct4- clusters (a cluster is defined as more than a single node)
Oct4-_size_avg – the average radius of the Oct4- clusters
Oct4-_size_std – the standard deviation in the radius of the Oct4- clusters
Oct4-_nd_cnt_avg – the average number of nodes of the Oct4- clusters
Oct4-_nd_cnt_std – the standard deviation in the number of nodes of the Oct4- clusters
Oct4-_rad_dist_avg – the average radial distance of the Oct4- clusters
Oct4-_rad_dist_std – the standard deviation of the radial distances of the Oct4- clusters
Total_obj_# - the total number of cells in the system
Object_#_ Oct4+ – the total number of Oct4+ cells in the system
Object_#_ Oct4- – the total number of Oct4- cells in the system
Percent_diff – the total number of Oct4- cells / the total number of cells
Agg_radius – the maximal radius (size) of the aggregate as measured from the center

Multiclass Metrics (dlx3b+, pSmad+, blx3b+/pSmad+):

dlx3b+_clust_# - the number of dlx3b+ clusters
dlx3b+_size_avg – the average radius of the dlx3b+ clusters
dlx3b+_size_std – the standard deviation in the radius of the dlx3b+ clusters
dlx3b+_nd_cnt_avg – the average number of nodes of the dlx3b+ clusters
dlx3b+_nd_cnt_std – the standard deviation in the number of nodes of the dlx3b+ clusters
dlx3b+_rad_dist_avg – the average radial distance of the dlx3b+ clusters
dlx3b+_rad_dist_std – the standard deviation of the radial distances of the dlx3b+ clusters
dlx3b+_clust_circ_avg – the average circularity of the dlx3b+ clusters
dlx3b+_clust_circ_std – the standard deviation of the circularities of the dlx3b+ clusters
dlx3b+_ecc_avg – the average eccentricity of the dlx3b+ clusters
dlx3b+_ecc_std – the standard deviation of the eccentricities of the dlx3b+ clusters
pSmad+_clust_# - the number of pSmad+ clusters
pSmad+_size_avg – the average radius of the pSmad+ clusters
pSmad+_size_std – the standard deviation in the radius of the pSmad+ clusters
pSmad+_nd_cnt_avg – the average number of nodes of the pSmad+ clusters

pSmad+_nd_cnt_std – the standard deviation in the number of nodes of the pSmad+ clusters
pSmad+_rad_dist_avg – the average radial distance of the pSmad+ clusters
pSmad+_rad_dist_std – the standard deviation of the radial distances of the pSmad+ clusters
pSmad+_clust_circ_avg – the average circularity of the pSmad+ clusters
pSmad+_clust_circ_std – the standard deviation of the circularities of the pSmad+ clusters
pSmad+_ecc_avg – the average eccentricity of the pSmad+ clusters
pSmad+_ecc_std – the standard deviation of the eccentricities of the pSmad+ clusters
pSmad+/dlx3b+_clust_# - the number of pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_size_avg – the average radius of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_size_std – the standard deviation in the radius of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_nd_cnt_avg – the average number of nodes of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_nd_cnt_std – the standard deviation in the number of nodes of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_rad_dist_avg – the average radial distance of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_rad_dist_std – the standard deviation of the radial distances of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_clust_circ_avg – the average circularity of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_clust_circ_std – the standard deviation of the circularities of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_ecc_avg – the average eccentricity of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_ecc_std – the standard deviation of the eccentricities of the pSmad+/ dlx3b+ clusters
pSmad+/dlx3b+_r_ratio – r_clust_nd_count_avg / y_clust_nd_count_avg
pSmad+/dlx3b+_g_ratio - r_clust_nd_count_avg / y_clust_nd_count_avg
Total_obj_# - the total number of cells in the system
Object_#_pSmad+/dlx3b+ – the total number of pSmad+/dlx3b+ cells in the system
Object_#_dlx3b+ – the total number of dlx3b+ cells in the system
Object_#_ pSmad+ – the total number of pSmad+ cells in the system
Agg_radius – the maximal radius (size) of the aggregate as measured from the center

Binary Metrics (Mesenchymal, Epithelial):
M_clust_# - the number of mesenchymal clusters
M_size_avg – the average radius of the mesenchymal clusters
M_size_std – the standard deviation in the radius of the mesenchymal clusters
M_nd_cnt_avg – the average number of nodes of the mesenchymal clusters
M_nd_cnt_std – the standard deviation in the number of nodes of the mesenchymal clusters
M_rad_dist_avg – the average radial distance of the mesenchymal clusters
M_rad_dist_std – the standard deviation of the radial distances of the mesenchymal clusters
E_clust_# - the number of epithelial clusters
E_size_avg – the average radius of the epithelial clusters
E_size_std – the standard deviation in the radius of the epithelial clusters
E_nd_cnt_avg – the average number of nodes of the epithelial clusters
E_nd_cnt_std – the standard deviation in the number of nodes of the epithelial clusters
E_rad_dist_avg – the average radial distance of the epithelial clusters
E_rad_dist_std – the standard deviation of the radial distances of the epithelial clusters
Total_obj_# - the total number of cells in the system

Object_#_E – the total number of mesenchymal cells in the system
Object_#_M – the total number of epithelial cells in the system
Percent_diff – the total number of epithelial cells / the total number of cells
Agg_radius – the maximal radius (size) of the aggregate as measured from the center

Supplementary Figures:

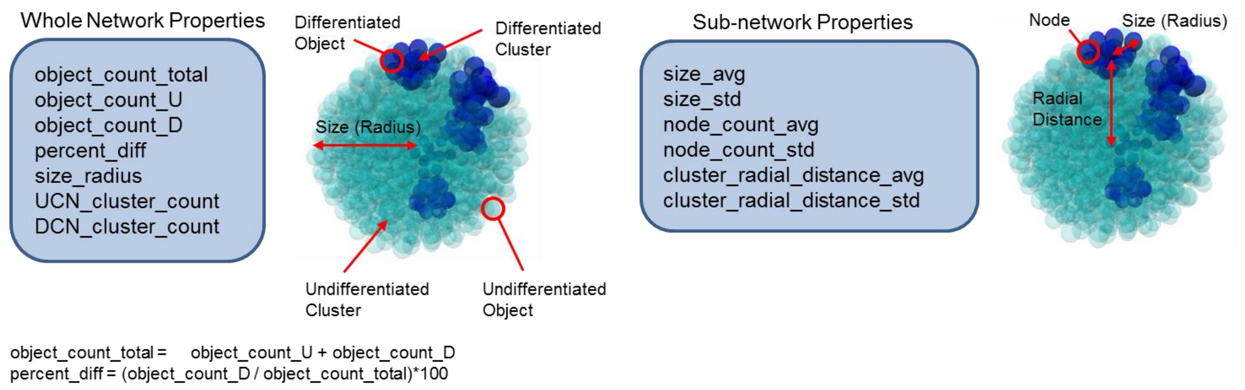

Figure S1 – Network metrics for characterizing spatial pattern formation and evolution. The whole network and sub network properties are outlined and annotated to give a visual representation of what each metric represents. In this example, U refers to undifferentiated Oct4+ cells, while D refers to more differentiated Oct4- cells.

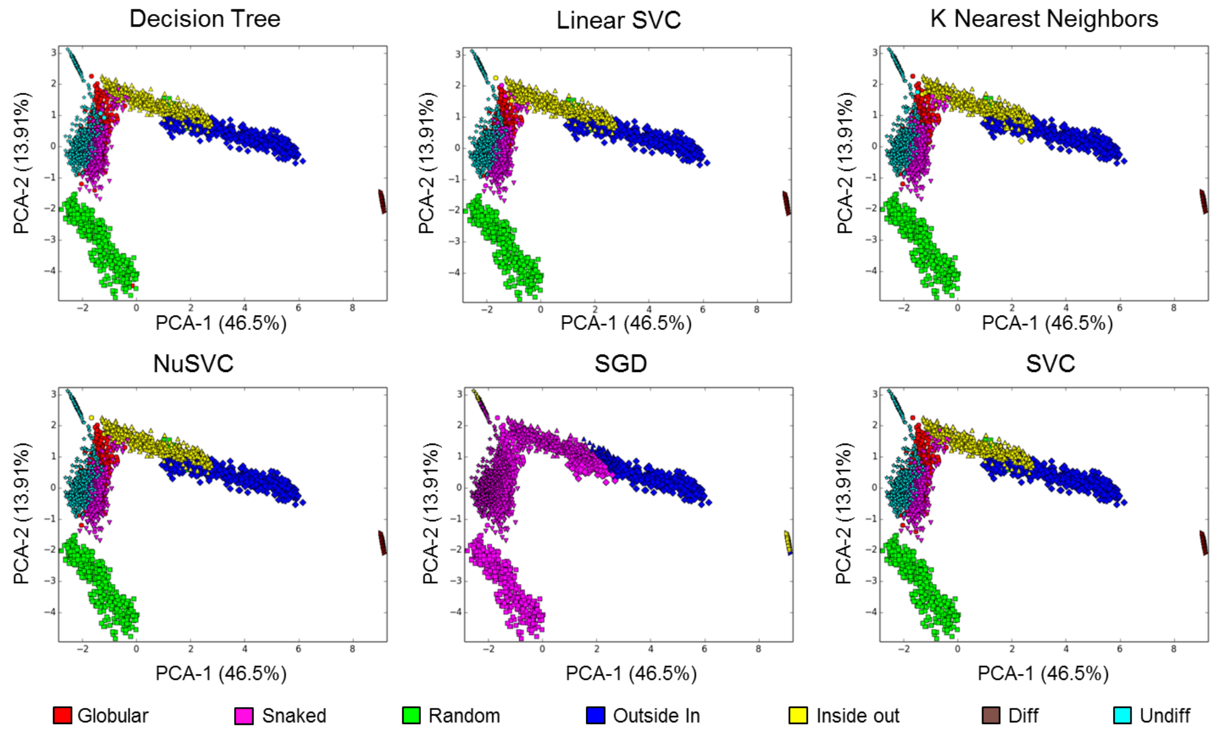

Figure S2 – A comparison of classification methods for correctly identifying patterns indicates that all classification schemes except stochastic gradient descent (SGD) recapitulate the true data. Colors indicate the different class of pattern assigned to the given data point.

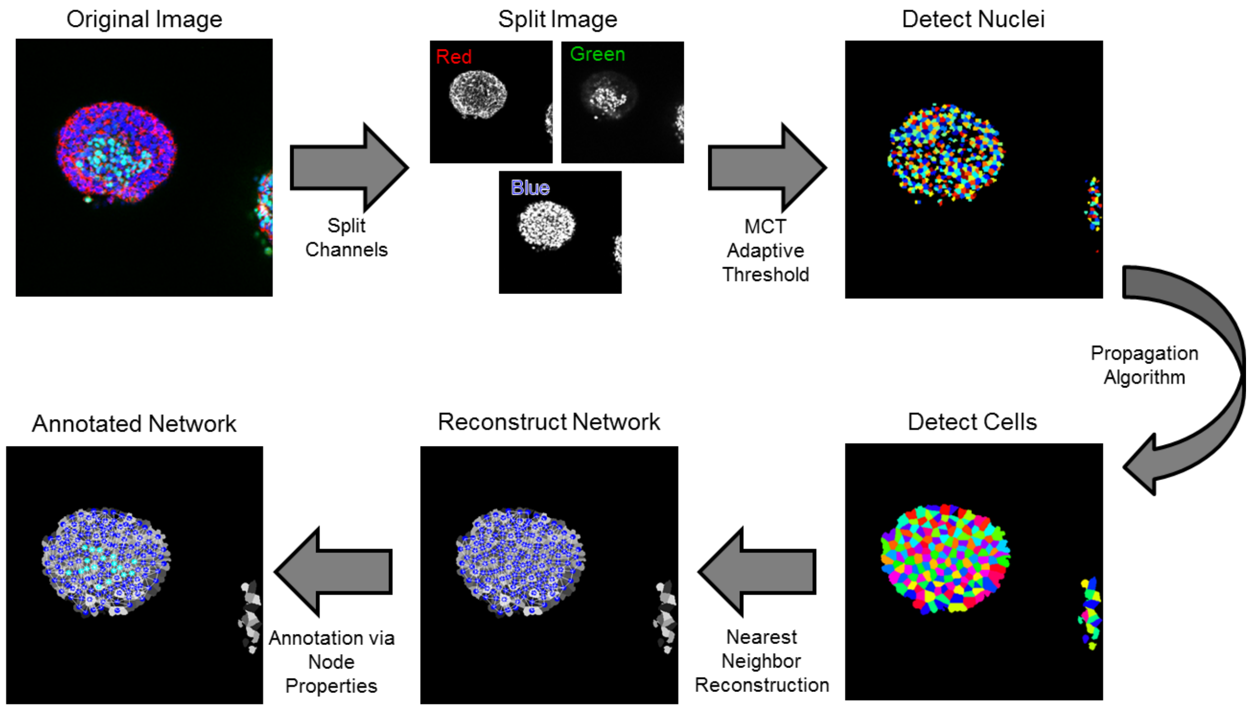

Figure S3 – Network digitization process for cellular aggregate images. The process of splitting images and detecting cells was performed in Cell Profiler, while the subsequent network reconstruction and annotation steps are performed using python.

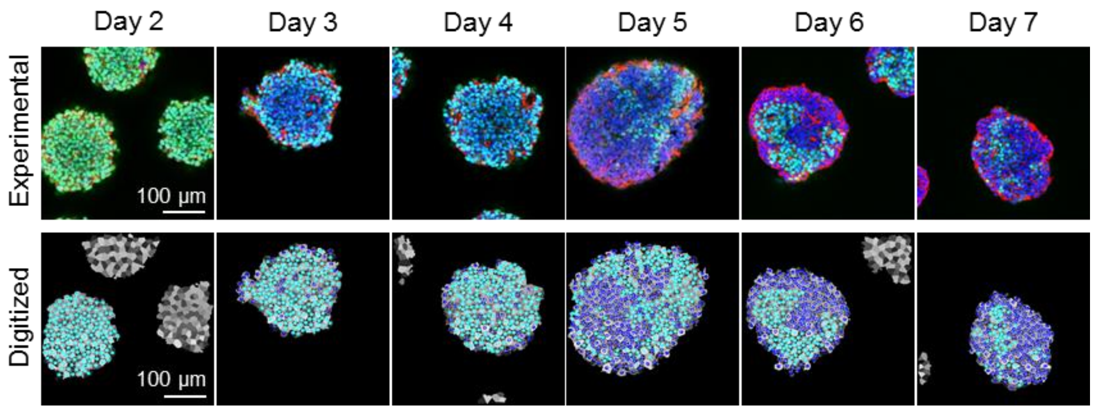

Figure S4 - Fidelity of network digitization process from experimental images (top row; Oct4 = green, DAPI = blue, phallodin = red) to reconstructed networks (bottom row; Oct4+ = teal, Oct4 - = blue). Scale bars = 100 μms.

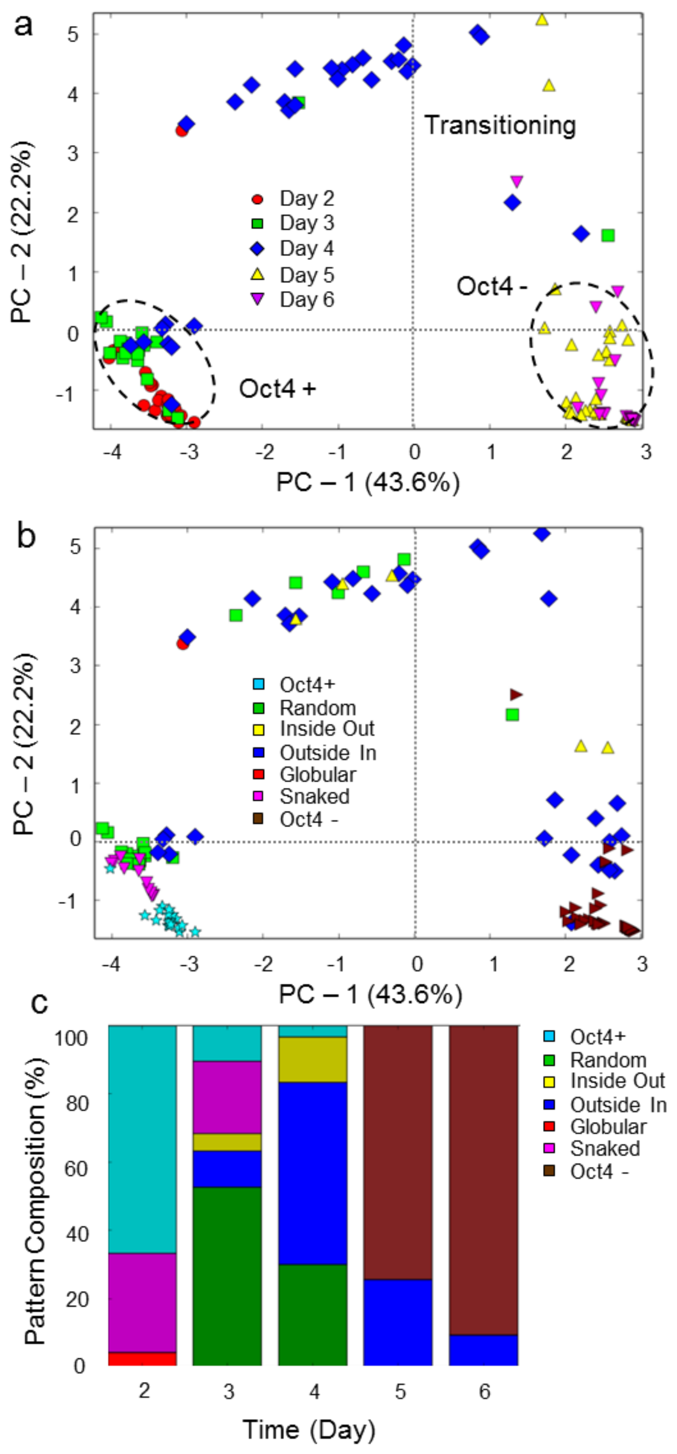

Figure S5 – 250 cell/aggregate pattern trajectory. **(a)** The differentiation trajectory where each point represents an individually analyzed network. The dashed regions denote which state the

observation falls into. **(b)** Individual networks color-coded by pattern type. **(c)** Pattern distributions over the 5 day differentiation time course.

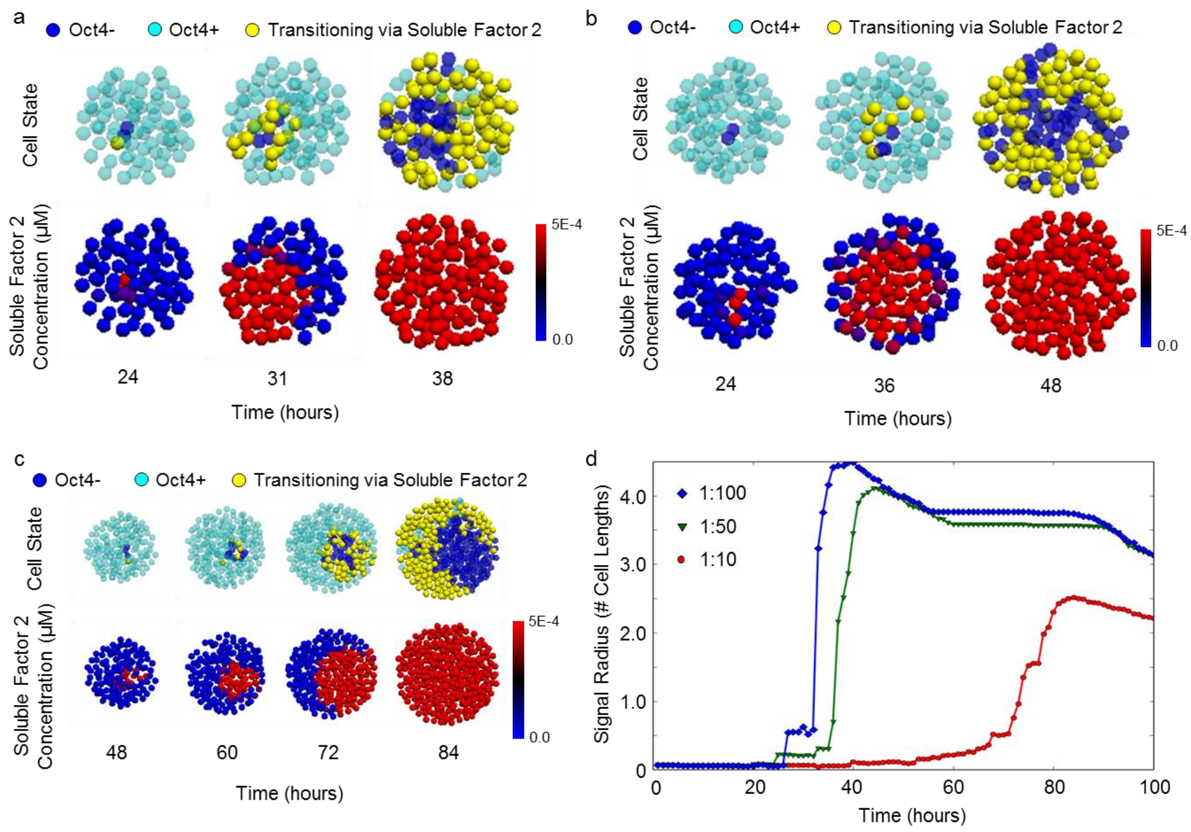

Figure S6 – Investigation of paracrine consumption/production ratios on the overall reach of a soluble paracrine gradient. Representative transition trajectories for the 1:100 **(a)**, 1:50 **(b)** and 1:10 **(c)** consumption to production ratios. **(d)** The average cell length influenced by the paracrine gradient for the 1:100 (blue), 1:50 (green) and 1:10 (red) consumption to production ratios.

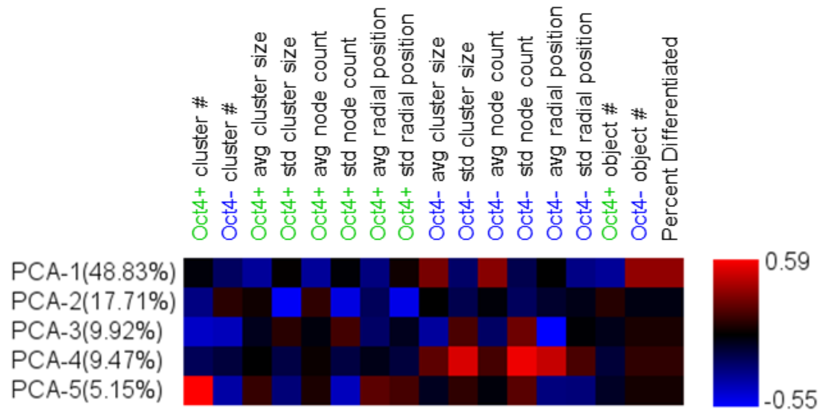

Figure S7 – Principal component analysis (PCA) for picking axes of largest variance. First 5 principal component axes together explained ~ 90% of the variance present in the data. Red and blue indicate a positive or negative contribution respectively of the metric with the axes of variation.

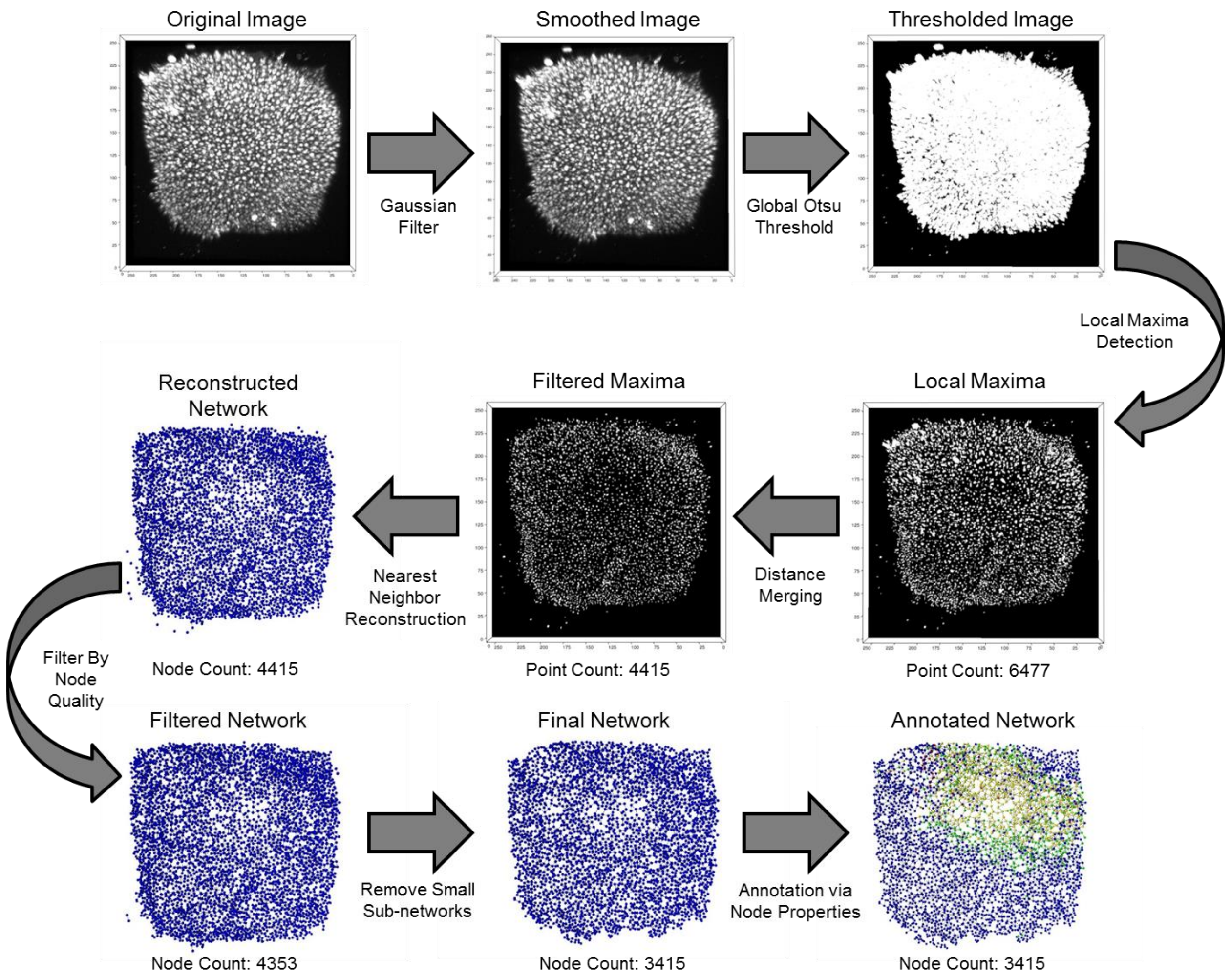

Figure S8 – Network digitization procedure for 3D confocal cichlid images. A Gaussian smoothing filtered was applied to filter out noise in the image. Next the image was thresholded using a local Otsu method. Next local maxima were detected using a local maxima filter. A simple thresholding and watershedding segmentation approach is used to separate individual cells using the maxima points as seeds. Next the network is reconstructed using a KD-tree method to infer cell connections. Final networks were generated by filtering out small unconnected nuclei.

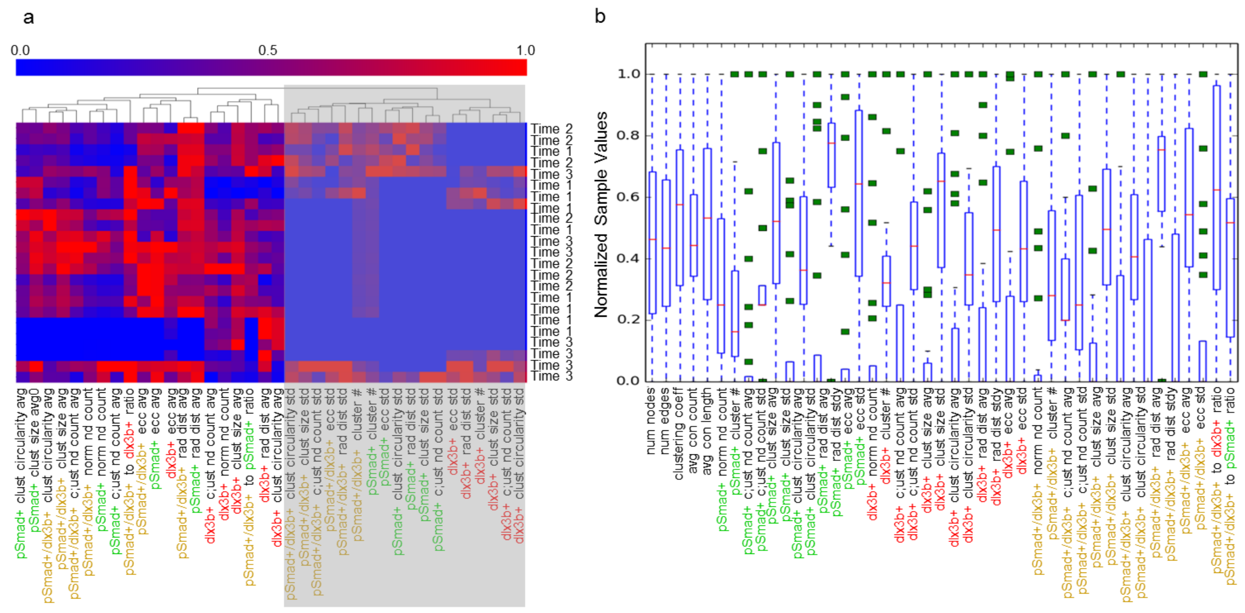

Figure S9 – **(a)** Investigating clustering of metrics for metric elimination. The gray region highlights metrics that all clustered together and were related to the standard deviations in measured values. Data were normalized by each metric such that the max value along each metric was 1 (red), while the minimum value was 0 (blue). **(b)** Identification of informative features by examination of metric distributions. Red lines represent the means, while blue boxes represent the first quartile, blue dashed lines represent the second quartile, and green squares represent outliers.

Supplementary Tables

Supplementary Table 1:

Rule Set	Parameter Name	Range
Local Random	α	0.001,0.0025, 0.005,0.0075, 0.01
Local Negative Feedback	α k1 n1	0.001, 0.005, 0.01 0.1,0.3, 0.5, 0.7, 0.9 10,50
Local Positive	α k1	0.001, 0.005, 0.01 0.1, 0.3, 0.5, 0.7, 0.9

Feedback	n1	10,50
Local Competing Feedback	α	0.001, 0.005, 0.01
	k1	0.1, 0.3, 0.6, 0.9
	n1	10,50
	k2	0.1, 0.3, 0.6, 0.9
	n2	10,50
Paracrine Positive	α	0.001, 0.005
	k1	0.006, .008, .01, .012, .014
	n1	10, 50
	p/q	10, 50, 100
Paracrine Negative	α	0.001, 0.005
	k1	0.006, .008, .01, .012, .014
	n1	10, 50
	p/q	10, 50, 100
Paracrine Competing	α	0.001, 0.005
	k1	0.006, .008, .01, .012, .014
	n1	10
	k2	.006, .008, .01, .012, .014
	n2	10
	p/q	10, 50, 100