scientific reports

OPEN

Deep learning of material transport in complex neurite networks

Angran Li[1], Amir Barati Farimani[1,2,3,4] & Yongjie Jessica Zhang[1,2]✉

Neurons exhibit complex geometry in their branched networks of neurites which is essential to the function of individual neuron but also brings challenges to transport a wide variety of essential materials throughout their neurite networks for their survival and function. While numerical methods like isogeometric analysis (IGA) have been used for modeling the material transport process via solving partial differential equations (PDEs), they require long computation time and huge computation resources to ensure accurate geometry representation and solution, thus limit their biomedical application. Here we present a graph neural network (GNN)-based deep learning model to learn the IGA-based material transport simulation and provide fast material concentration prediction within neurite networks of any topology. Given input boundary conditions and geometry configurations, the well-trained model can predict the dynamical concentration change during the transport process with an average error less than 10% and 120 $\sim$ 330 times faster compared to IGA simulations. The effectiveness of the proposed model is demonstrated within several complex neurite networks.

The geometry of neurites is known to exhibit complex morphology which is essential for neuronal function and biochemical signal transmission. However, highly branched networks of neurites also make it challenging to mediate intracellular material transport because material synthesis and degradation in neurons are carried out mainly in the cell body[1, 2] which leads to a long-distance transport for the material. The disruption of this long-distance transport can induce neurological and neurodegenerative diseases like Huntington's, Parkinson's and Alzheimer's disease[3–7]. Therefore, it has attracted a considerable amount of attention in recent years to study transport mechanisms and build mathematical transport models in neurite networks. Recent studies show that molecular motors play fundamental roles in intracellular transport to carry the material and move directionally along the cytoskeletal structure like microtubules or actin filaments[8–10].

Motivated by these findings, different mathematical models based on partial differential equations (PDEs) have been proposed to help understand transport mechanisms and pathology of neuron diseases. For instance, Smith and Simmons developed a generic model of molecular motor-assisted transport of cell organelles and vesicles along filaments[11]. Based on this motor-assisted transport model, Friedman and Craciun presented an axonal transport model by considering the ability of material to bind more than one motor protein[12]. Craciun et al. introduced pausing of the material during the transport process in the model to study slow axonal transport[13]. In addition, several PDE models were developed to study transport impairment by accounting for traffic jam[14] and microtubule swirls[15] in abnormal neurons. Though the aforementioned models provide a reasonable mechanistic explanation of transport mechanism or formation of the transport impairment in neurites, most of these models were solved only in one-dimensional (1D) domain without considering the impact of complex neurite geometry. Recent advances in numerical methods and computing resources allow us to simulate the detailed cellular process in complex geometry using 1D or three-dimensional (3D) PDE models. For instance, computational software based on finite element method (FEM) has been used to solve PDE models in neuron[16] and cell biological systems[17]. The FEM approach was also used to model the extracellular electrical neural microstimulation and presented a clearer difference of neuron response compared to the hybrid FEM-cable-equation approach[18]. However, these tools have accuracy and computational cost issues when tackling highly branched geometry like neurite networks. Based on the conventional FEM, isogeometric analysis (IGA)[19] was proposed to directly integrate geometric modeling with numerical simulation and achieve better accuracy and robustness compared to FEM. With the advances in IGA, one can simulate the transport process by solving PDEs in an accurately reconstructed neuron geometry. In our previous study, we developed an IGA-based simulation platform to solve a 3D motor-assisted transport model in several complex neurite networks and obtain the spatiotemporal material

[1]Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. [2]Department of Biomedical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. [3]Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. [4]Department of Machine Learning, Carnegie Mellon University, Pittsburgh, PA, USA. ✉email: jessicaz@andrew.cmu.edu

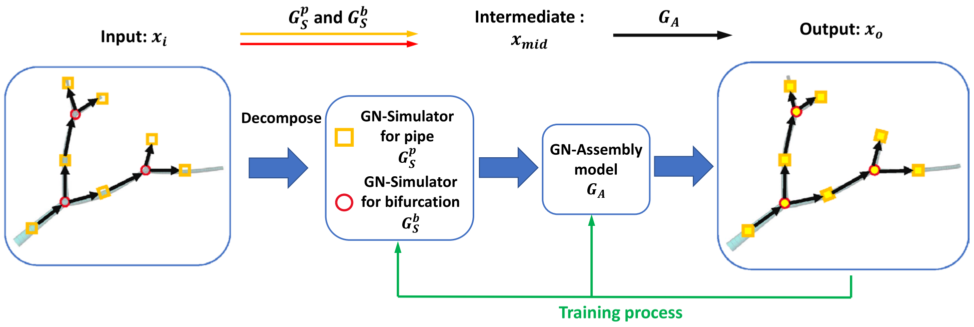

Figure 1. An overview of the GNN model to learn and predict neuron material transport process. The input neurite network is first decomposed into pipes and bifurcations to create the graph representation of the neurite network. Next, the input features x_i of each pipe or bifurcation are processed by the corresponding GNN simulator (G_S^p or G_S^b) to generate intermediate concentration result x_{mid}. Then, the GNN assembly model (G_A) takes x_{mid} as input and computes the interaction between simulators to predict concentration result x_o.

distribution[20]. However, the high computational cost to perform 3D simulations has limited its application in the biomedical field when fast feedback from computer simulation is needed.

To address limitations in the current simulation platform, we propose to build a surrogate model by combining deep learning (DL) with IGA simulation. DL has been proven successful in computer vision and image recognition[21–23] by handling a large number of labeled datasets and providing fast end-to-end prediction. The practical success of DL in artificial intelligence also inspires its application in solving high-dimensional PDEs[24] and learning the physics behind PDE models[25]. In particular, deep neural networks (DNNs) are becoming popular in surrogate modeling because it can be far more efficient when predicting complex phenomena[26]. For instance, Farimani et al. applied conditional generative adversarial networks (*cGAN*) in a data-driven paradigm for rapid inference, modeling and simulation of transport phenomena[27]. Wiewel et al. proposed a DL-based fluid simulator to predict the changes of pressure fields over time[28]. Li et al. developed an encoder-decoder based convolutional neural network (CNN) to directly predict concentration distribution of a reaction–diffusion system, bypassing the expensive FEM calculation process[29]. While these works manage to learn the underlying physical models for prediction, they are limited to handle the problem in relatively simple geometry with Euclidean data (e.g. structured grid) available for training. Recently, many studies on graph neural networks (GNNs) have emerged to extend DL techniques for data defined in the graph structure. Inspired by the success of CNNs[21], a large number of methods were developed to re-define the convolutional operation for graph data and achieve great performance in computer vision tasks like graph node classification and image graph classification[30–33]. GNNs were also applied to predict the drag force associated with the laminar flow around airfoils[34] or the property of crystalline materials[35], understand the interaction behind physics scenes[36], solve PDEs by modeling spatial transformation functions[37], or learn particle-based simulation in complex physical systems[38].

In this study, we develop a GNN-based model to learn the material transport mechanism from simulation data and provide a fast prediction of the transport process within complex geometry of neurite networks. The use of GNN is motivated by the extensive topologies of neurite networks and IGA simulation data stored in the mesh structure. Moreover, the GNN model can achieve better computational efficiency than IGA simulation without sacrificing too much geometry information of neuron. For instance, our GNN model can preserve the spatial concentration distribution on the cross section of 3D neuron geometries. To ensure the model is applicable to any neurite geometry, we build a graph representation of neurons by decomposing the neuron geometry into two basic structures: pipe and bifurcation. Different GNN simulators are designed for these two basic structures to predict the spatiotemporal concentration distribution given input simulation parameters and boundary conditions. Specifically, we add the residual terms from PDEs to instruct the model to learn the physics behind simulation data. To reconstruct the neurite network, a GNN-based assembly model is used to combine all the pipes and bifurcations following the graph representation. The loss function of the assembly model is designed to impose consistent concentration results on the interface between pipe and bifurcation. The well trained GNN model can provide a fast and accurate prediction of the material concentration distribution, leading to an efficient DL framework for studying the transport process in complex 3D models of neurite network. The framework was applied to several complex neurite networks achieving an average error less than 10% and 120 $\sim$ 330 times faster compared to IGA simulations.

Results

GNN model overview. The aim of our GNN model is to learn from the material transport simulation data and predict the transport process in any neurite network. However, the geometry diversity of neurite networks makes it impossible to train the DL model directly on the entire neurite network. To address this issue, we introduce a graph representation of the neurite network and build the GNN model based on the graph network (GN) framework[36]. A neurite network can be decomposed into two basic structures: pipe and bifurcation (yellow square and red circle in Fig. 1, respectively). Each structure can be treated as one node in the graph and the nodes can be connected following the skeleton of the neurite network to constitute the graph. Based on the

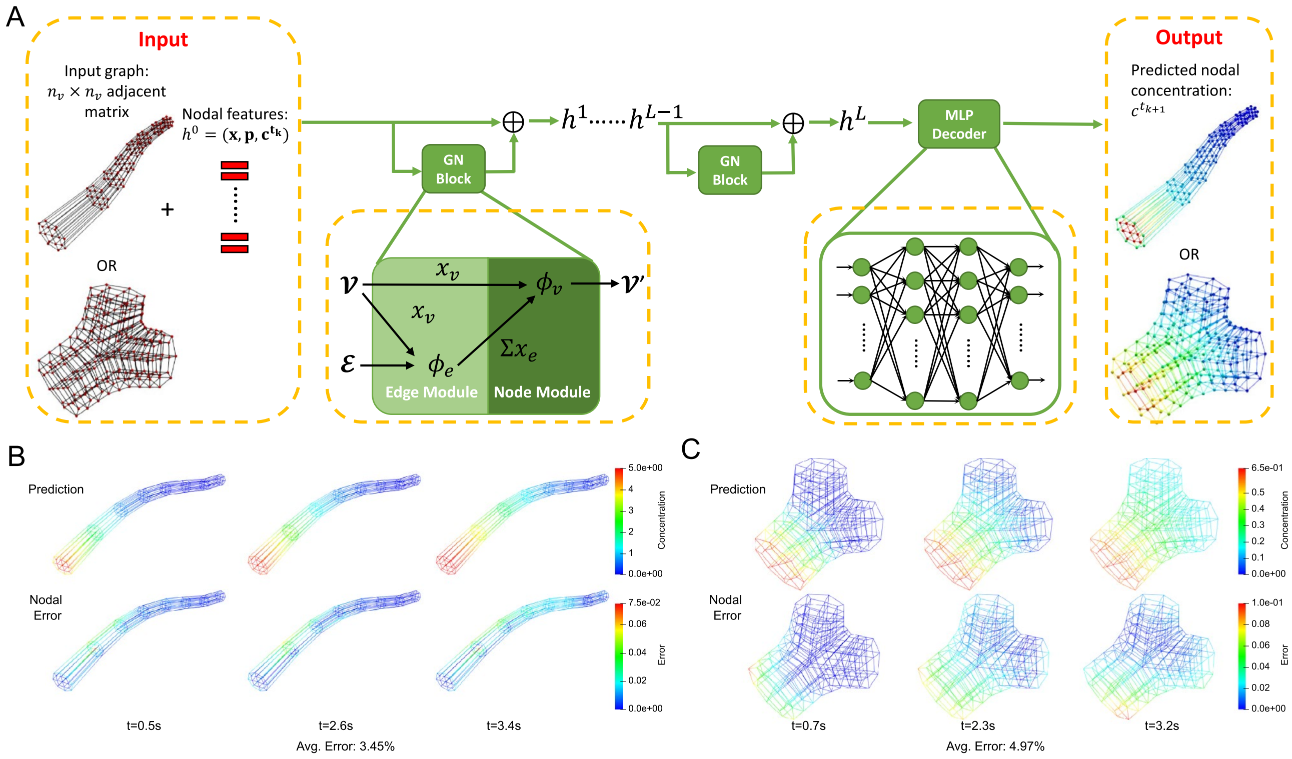
A
Input
Input graph: $n_v \times n_v$ adjacent matrix
Nodal features: $h^0 = (\mathbf{x}, \mathbf{p}, \mathbf{c}^{t_k})$
OR
GN Block
$h^1 \ldots\ldots h^{L-1}$
h^L
MLP Decoder
Edge Module
Node Module
Output
Predicted nodal concentration: $c^{t_{k+1}}$
B
C
Prediction
Nodal Error
Concentration
Error
t=0.5s
t=2.6s
t=3.4s
Avg. Error: 3.45%
t=0.7s
t=2.3s
t=3.2s
Avg. Error: 4.97%

Figure 2. Results of GNN simulators. (**A**) The architecture of the GNN simulators adopts a recurrent "GN Block + MLP Decoder" scheme on the input graph to compute nodal concentration values at each time step from input nodal features. The input n_v-node graph is stored in a $n_v \times n_v$ adjacent matrix. The nodal features include coordinate vector x, simulation parameter vector p and initial concentration values c^{t_k} at time step t_k. The L-step "GN Block" follows Algorithm 1 to compute the interaction among nodes and generates a series of updated latent graphs with hidden nodal embeddings, h^1, ..., h^L. The "MLP Decoder" outputs nodal concentration values $c^{t_{k+1}}$ at the next time step from the nodal embedding h^L of the final latent graph. (**B**, **C**) The concentration distribution comparison of pipe and bifurcation simulators, respectively. For each simulator, we plot the prediction and nodal error results at three different time steps. The average errors are 3.45% and 4.97% for the pipe and bifurcation simulators, respectively.

graph representation of the neurite network, two separate GNN simulators are built for the pipe (G_S^p) and the bifurcation (G_S^b), respectively. Given input features x_i including node locations, simulation parameters and initial nodal concentration, the simulators can output the intermediate nodal concentration result x_{mid} in the pipe and the bifurcation, respectively. To obtain a consistent global concentration result, a GNN assembly model (G_A) is used to learn the interaction between different structures so that given intermediate value x_{mid}, the model can assemble different structures and output the final prediction x_o on the graph. In sum, our GNN model consists of

- The GNN simulators for local prediction in pipe and bifurcation structures; and
- The GNN assembly model for global prediction in the entire neurite network.

We implement our GNN model using PyTorch[39] and the "PyTorch Geometric" library[40]. The detailed results will be explained in the following sections.

GNN simulators for pipe and bifurcation.

Since the pipe and the bifurcation have different geometry topologies, we train two separate simulators to handle different graph structures extracted from the simulation results. Both simulators share the same recurrent "GN block + MLP Decoder" architecture but are trained with pipe and bifurcation datasets, respectively. As shown in Fig. 2, the input feature vectors depicting geometry, parameter settings, boundary conditions and concentration values c^{t_k} at t_k are first processed by a series of GN blocks to learn the hidden layer representations that encode both local graph structure and features of nodes. In our GNN simulator, the GN block includes two modules ϕ_e, ϕ_v to update the edge and node features, respectively. ϕ_e computes the concentration gradient along the edge and the length of the edge. Given the spatial coordinates $\boldsymbol{p}_i$, $\boldsymbol{p}_j$ and concentration c_i, c_j of two end nodes on edge e_{ij}, ϕ_e outputs the edge attributes $\boldsymbol{x}_{e_{ij}}$ as $\left[(c_j - c_i)/||\boldsymbol{p}_j - \boldsymbol{p}_i||, ||\boldsymbol{p}_j - \boldsymbol{p}_i||\right]$. ϕ_v is a multilayer perceptron (MLP) consisting of two hidden layers (with ReLU activations), each layer with the size of 32. The forward computation of our GN block is characterized by Algorithm 1. After the GN blocks output the latent graph that encodes all the node features, a MLP is then used to decode the hidden layer representation and output predicted concentration values $c^{t_{k+1}}$ at the next time step t_{k+1}.

The MLP has three hidden layers (with ReLU activations), followed by a non-activated output layer, each layer with the size of 32.

```
Algorithm 1: GN Block
  Input: Graph, $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \boldsymbol{x}_v)$
  for each edge $e_{ij} \in \mathscr{E}$ do
      Compute edge attributes: $\boldsymbol{x}_{e_{ij}} = \phi_e(\boldsymbol{x}_{v_i}, \boldsymbol{x}_{v_j})$;
  end
  for each node $v_i \in \mathscr{V}$ do
      Aggregate edge attributes: $\boldsymbol{x}_{e,v_i} = \sum_j \boldsymbol{x}_{e_{ij}}$;
      Compute output: $\boldsymbol{x}'_{v_i} = \phi_v(\boldsymbol{x}_{v_i}, \boldsymbol{x}_{e,v_i})$;
  end
  Output: Graph, $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \boldsymbol{x}'_v)$
```

The input graph of the simulator is created by extracting certain nodes from each cross section of a hexahedral mesh following the templates (Supplementary Fig. S1). For each pipe structure, we use a circular plane template (Supplementary Fig. S1C) to extract 17 nodes from each cross section. For each bifurcation structure, we use another template (Supplementary Fig. S1D) to extract 23 nodes from the cross section at the bifurcation branch point. We also use the same circular plane template to extract 3 circular cross sections on each branch of the bifurcation. For each node, we collect the geometry information and simulation parameters in an input feature vector. Here, the geometry information of each node is encoded by its coordinates and the radius of the cross section on which the node is located. The nodal concentration value is set to be the target prediction.

To encourage the model to learn the physics underlying the simulation data, we add the residuals of the governing equation (Eq. 3 in Methods) to the mean squared error (MSE) loss function as

$$\mathscr{L}_{simulator} = \frac{1}{N}\sum_{i=1}^{N}\left[(c_{0,i}^P - c_{0,i}^G)^2 + (c_{\pm,i}^P - c_{\pm,i}^G)^2\right] + \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\partial c_{0,i}}{\partial t} - D\nabla^2 c_{0,i} + (k_+ + k_-)c_{0,i} - k'_+ c_{+,i} - k'_- c_{-,i}\right]^2 + \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\partial c_{\pm,i}}{\partial t} + \boldsymbol{u}_\pm \cdot \nabla c_{\pm,i} - k_\pm c_{0,i} + k'_\pm c_{\pm,i}\right]^2, \tag{1}$$

where c_0, c_+ and c_- are the spatial concentrations; D is the diffusion coefficient of materials; $\boldsymbol{u}_+$ and $\boldsymbol{u}_-$ are velocities of materials; $k_\pm$ and $k'_\pm$ are filament attachment and detachment rates (See Eq. 3 in Methods for the detailed physical meaning of the aforementioned variables); N denotes the number of nodes on the graph; and superscripts P and G denote the prediction and the ground truth value on the graph, respectively.

To create the training dataset for two simulators, we first use our IGA solver to run the material transport simulation (see Methods for details) in two complex zebrafish neurons from the NeuroMorpho.Org database[41] (NMO_66731 and NMO_66748 in Fig. 3B,D). Regarding the simulation setting, we use 200 different boundary condition values and set constant parameters in Eq. (3) and Eq. (5) as $D = 1.0\ \mu m^2/s, k_\pm = 1.0\ s^{-1}, k'_\pm = 0.5\ s^{-1}$ and $u_i = 0.1\ \mu m/s$. The time step for each simulation is set to be $0.1\ s$. We simulate until the transport process is steady and then extract the spatiotemporal simulation results of 100 different geometries for each simulator from these two neurons. Note that for each geometry, we extract all the time-sequence simulation results until the steady state point as the training data. The nodal feature vectors and nodal concentration values are stored for each geometry to build the training dataset that contains 20,000 samples for each simulator. After establishing the architecture of two simulators, we train each simulator by randomly selecting 75% samples as the training data. The Adam optimizer[42] is used to optimize the model with the step learning rate decay ranging from 10^{-3} to 10^{-6}. Our simulators are trained for 200 epochs and we evaluate the performance of the model using the rest 25% samples as the test dataset. At the end of training, the test loss for the pipe and bifurcation simulators converges to 0.265 and 0.111, respectively (Supplementary Fig. S2).

To demonstrate the performance of our GNN simulators, we select three prediction results of a pipe and a bifurcation from the test dataset and compare them with their corresponding IGA simulation results in Fig. 2. For the pipe simulator, we find the model can accurately capture the boundary conditions and predict the concentration distribution at each time step, which indicates that the GNN simulator manages to learn the time-dependent behavior of the transport equations. By comparing nodal error at $t = 2.6\ s$ and $t = 3.4\ s$ in Fig. 2B, we find the error increases along with the front of material propagation through the pipe, which suggests that the pipe simulator is sensitive to the sudden distribution change during the material propagation and needs further improvement. For the bifurcation simulator, the predicted result has higher accuracy around the branch region which suggests that the bifurcation simulator learns to transport the material in correct directions at the branch point. However, the boundary condition is not preserved as well as the pipe simulator performs. The possible reason is that the boundary nodes have fewer neighboring nodes for edge feature aggregation compared to the interior nodes and the lack of neighboring information leads to higher error. Therefore, our bifurcation simulator can be further improved to balance the neighborhood between the boundary and interior nodes to better preserve the boundary condition.

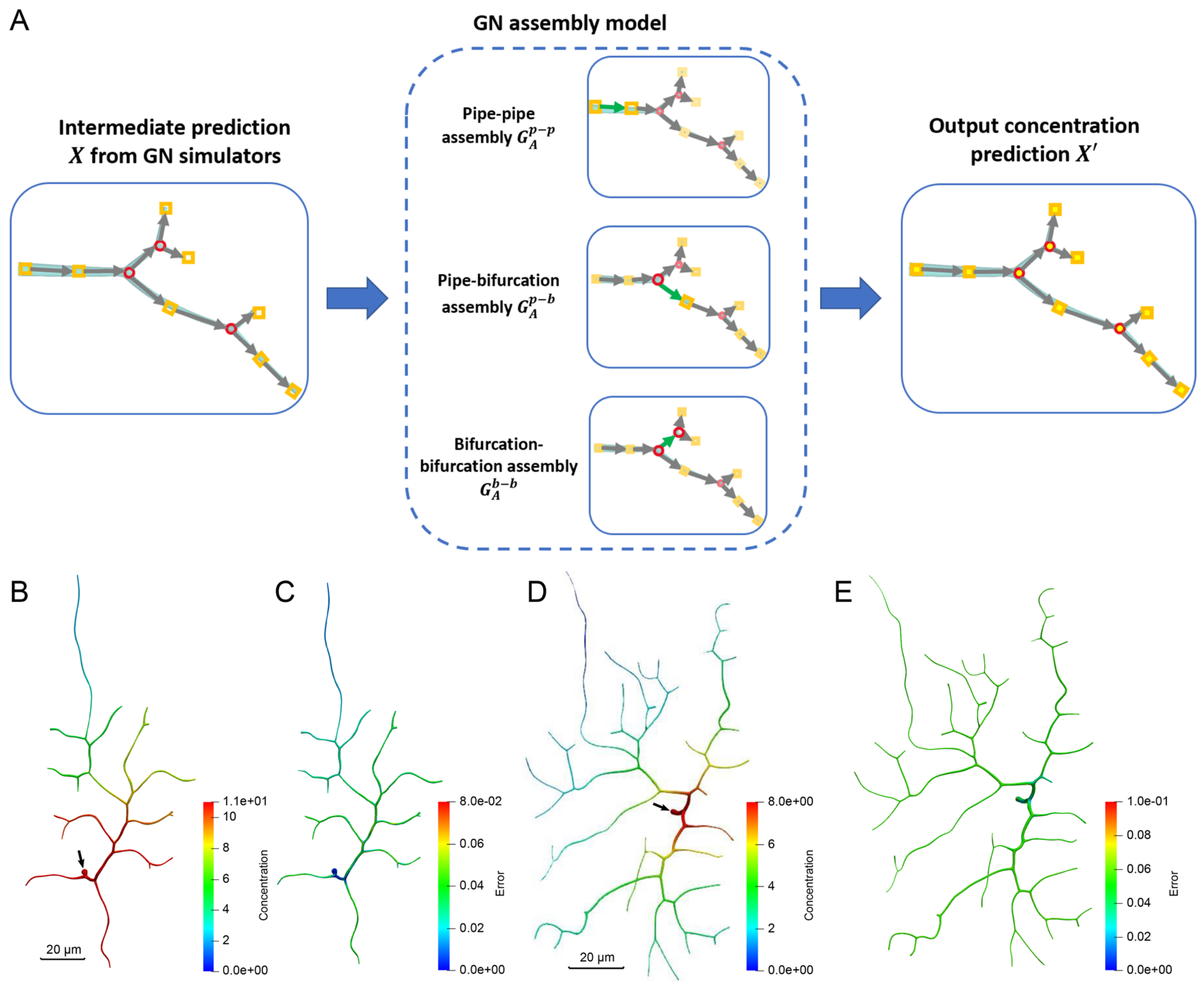
A
GN assembly model
Intermediate prediction X from GN simulators
Pipe-pipe assembly G_A^{p-p}
Pipe-bifurcation assembly G_A^{p-b}
Bifurcation-bifurcation assembly G_A^{b-b}
Output concentration prediction X'
B
C
D
E
20 μm
1.1e+01
10
8
6
4
2
0.0e+00
Concentration
8.0e-02
0.06
0.04
0.02
0.0e+00
Error
8.0e+00
6
4
2
0.0e+00
Concentration
20 μm
1.0e-01
0.08
0.06
0.04
0.02
0.0e+00
Error

Figure 3. Results of the GNN assembly model. (**A**) The GNN assembly model takes the intermediate prediction X from individual GNN simulators as input and outputs the final concentration prediction X'. The GNN assembly components G_A^{p-p}, G_A^{p-b} and G_A^{b-b} adopt the "message-passing" scheme to aggregate the intermediate prediction from the neighboring nodes to update the concentration prediction. The green arrows show the message-passing process during the assembly. (**B**, **C**) The predicted concentration and the nodal prediction error of NMO_66731 at steady state ($t = 25\ s$). (**D**, **E**) The predicted concentration and the nodal prediction error of NMO_66748 at steady state ($t = 31\ s$). In (**B**) and (**D**), the black arrows point to the inlet of material. Scalar bar: 20 μm. Unit for color bars: $mol/\mu m^3$.

We perform 4-fold cross validation and the mean relative error (MRE) results for both simulators are shown in the second column of Table 1. The well-trained GNN simulators can provide concentration prediction with an error of less than 8% on average. With a standard deviation less than 1.3%, the model shows good performance when being generalized to unknown data. To study the impact of the PDE residuals in Eq. (1) on the prediction performance, we compare our simulators with the model trained using the standard MSE loss function. We perform cross validation with the same dataset for each comparison and the results are shown in Table 1. The comparison shows that the model achieved better accuracy when trained with the "MSE + PDE residuals" loss function, which indicates the physics information contained in PDE residuals is learned by the model and improves its performance.

GNN assembly model. The objective of the GNN assembly model is to assemble local prediction from simulators and output an improved continuous concentration distribution on the entire geometry. An overview of the GNN assembly model is shown in Fig. 3. The assembly model needs to learn the interaction between two simulators. Here the assembly model includes three components to cover all the assembly scenario in a decomposed neurite network: (a) pipe and pipe G_A^{p-p}; (b) pipe and bifurcation G_A^{p-b}; and (c) bifurcation and bifurcation G_A^{b-b}. During assembly, the model loops each simulator node on the decomposed neurite network and utilizes the "message-passing" scheme to gather predicted results from its neighboring simulator nodes (green arrows in Fig. 3A). In particular, the nodal predicted results on the interface between two simulator nodes are collected.

Loss function	MSE	MSE + PDE residuals
Pipe simulator	10.9 ± 3.45%	6.10 ± 1.25%
Bifurcation simulator	13.5 ± 2.47%	7.20 ± 0.78%

Table 1. MRE comparison between simulators using different loss functions.

Then, all the collected values from their neighboring simulator nodes are concatenated with values from the current simulator node and processed by a MLP to improve the prediction. While G_A^{p-p}, G_A^{p-b} and G_A^{b-b} have different numbers of input and output values, they share the same MLP architecture with three hidden layers (with ReLU activations), followed by a non-activated output layer, each layer with the size of 32.

To ensure the concentration result is consistent on the interface between two simulators, we add a penalty term to the MSE loss function as

$$\mathscr{L}_{assembly} = \frac{1}{N}\sum_{i=1}^{N}\left\{(c_{0,i}^{P} - c_{0,i}^{G})^2 + (c_{\pm,i}^{P} - c_{\pm,i}^{G})^2\right\} + \alpha\frac{1}{M}\sum_{j=1}^{M}\left\{(c_{0,j}^{s_1,interface} - c_{0,j}^{s_2,interface})^2 + (c_{\pm,j}^{s_1,interface} - c_{\pm,j}^{s_2,interface})^2\right\}, \tag{2}$$

where N denotes the number of nodes on two assembled simulators, M denotes the number of nodes on the interface, superscript P and G denote the prediction and the ground truth value on the graph, respectively. Superscripts s_1 and s_2 denote the prediction value from the first and second simulators, respectively. α is the penalty strength which is set to be 10 in this study.

We use the same IGA simulation results in two complex geometries of zebrafish neurons (NMO_66731 and NMO_66748 in Fig. 3B,D) to create a training dataset for the assembly model. Based on the graph representation of these two trees, we follow three basic assembly structures and extract 30 different geometries for each type of assembly. The simulation results of these geometries are output and create 6,000 samples for each assembly structure. The model is trained using 75% samples as the training dataset and we evaluate the performance of the model using the rest 25% samples as the test dataset. At the end of training, the test loss for the GNN assembly model converges to 0.1492 (Supplementary Fig. S3). To test the performance of the assembly model, we use these two complex geometries and compare the concentration prediction at steady state (Fig. 3B–E). The prediction MREs are 6.7% for NMO_66731 and 7.3% for NMO_66748, respectively. We find that the assembly model manages to generate a continuous global concentration distribution for each geometry. The comparison of predicted concentration between nodal error for each geometry also shows that high concentration regions are predicted accurately. This indicates that our GNN model can capture the locations that are easier to develop transport disruption due to material accumulation.

Results for complex neurite networks. After the simulators and the assembly model in the GNN framework are well trained, we use our GNN model to predict the concentration distribution in several complex neurite networks and compare with the simulation results from our IGA solver. All complex neurite networks selected from the NeuroMorpho.Org database are shown in Figs. 4 and 5. Since the GNN model is trained with the simulation data in zebrafish neurons, we pick another two zebrafish neurons (Fig. 4A,B) to test the model performance in the neurons from the same species. We also pick another six mouse neurons (Fig. 4C,G,H, 5) from a different species to validate the model. We choose neurons with the number of bifurcations ranging from 9 to 356. For each neuron, we first run IGA simulation to get the ground truth concentration results and then compare with the predicted concentration obtained from our GNN model. Fig. 4 shows the prediction results for the neurons with longer branches and Fig. 5, shows neurons with more bifurcations and shorter branches. We observe that the model performance gets worse in longer branches or regions with a high density of bifurcations. The possible reason is that the increasing complexity of the geometry leads to a larger graph representation which affects the prediction accuracy due to the complicated assembling process on more simulators.

The detailed geometry information, computation statistics and error comparison are summarized in Table 2. By comparing the prediction MRE of zebrafish neurons (top four rows in Table 2), we find the MRE only increases by around 0.8% for all the tested zebrafish neurons, which validates the applicability of the model among neurons from the same species. In addition, we find that the average prediction MREs of zebrafish and mouse neurons are comparable with 7.18% and 8.23%, respectively, indicating that the trained model can also work for neurons from other species.

To evaluate the computation performance, we define a speedup ratio as the ratio between the IGA computation time and the GNN prediction time. We run the code and measure the computation time on the XSEDE (Extreme Science and Engineering Discovery Environment) supercomputer Bridges[43] in the Pittsburgh Supercomputer Center. For each geometry, we perform the IGA simulation through parallel computing on CPU and perform GNN prediction on GPU. The speedup ratio for each geometry is shown in Table 2 and we find that our GNN model can achieve up to 330 times faster compared to IGA simulation. We also observe that the speedup ratio decreases when the neuron geometry becomes complicated with more bifurcations. The ratio converges to around 120 when the bifurcation number reaches 356, which is still a significant improvement by reducing computation time from hours to minutes.

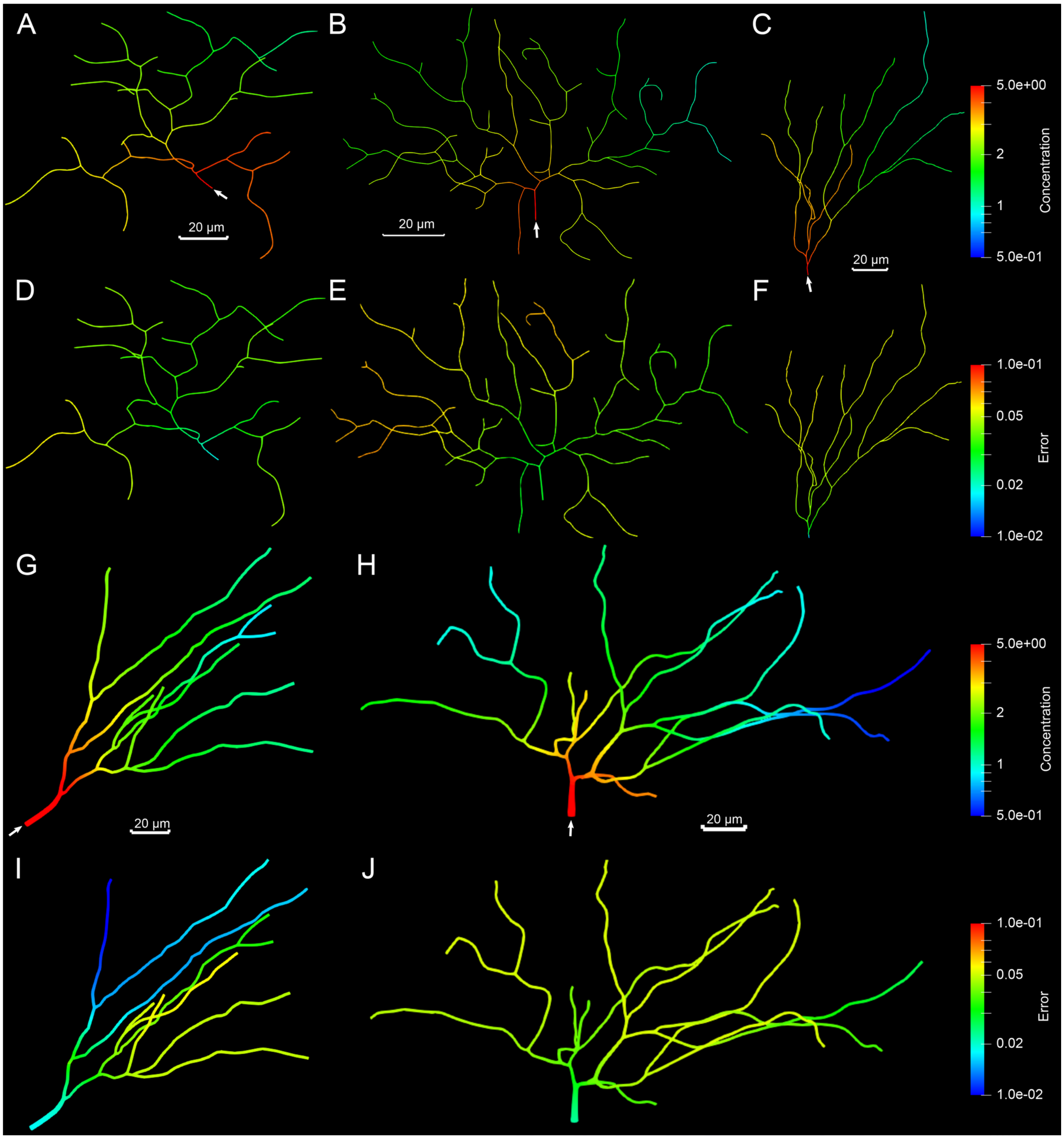

Figure 4. The concentration prediction of material transport in the neurite network geometry of NMO_06846, NMO_06840, NMO_112145, NMO_32235 and NMO_32280. (**A**–**C**, **G**, **H**) The predicted concentration results of steady state at $t = 15\ s, 26\ s, 19\ s, 28\ s$ and $32\ s$, respectively. The white arrows point to the inlet of material. Scalar bar: 20 μm. Unit for color bars: $mol/\mu m^3$. (**D**–**F**, **I**, **J**) The nodal errors between the ground truth and predicted concentration. Logarithmic scale is used to highlight the distribution pattern.

Discussion

In this paper, we present a GNN-based DL model for predicting the material transport process in complex geometry of neurons. Our strategy utilizes a graph representation of neurons to overcome the difficulty in handling different neuron geometries with the DL model. The underlying assumption of our approach is that the material concentration distribution can be predicted locally in simple geometries and then be assembled following the graph representation to restore the concentration distribution in any complex geometry. Two GNN models are developed to validate our assumption. The first GNN-based model serves as the simulator to predict dynamic concentration results locally on the mesh of two basic structures: pipe and bifurcation. We adopt GNN

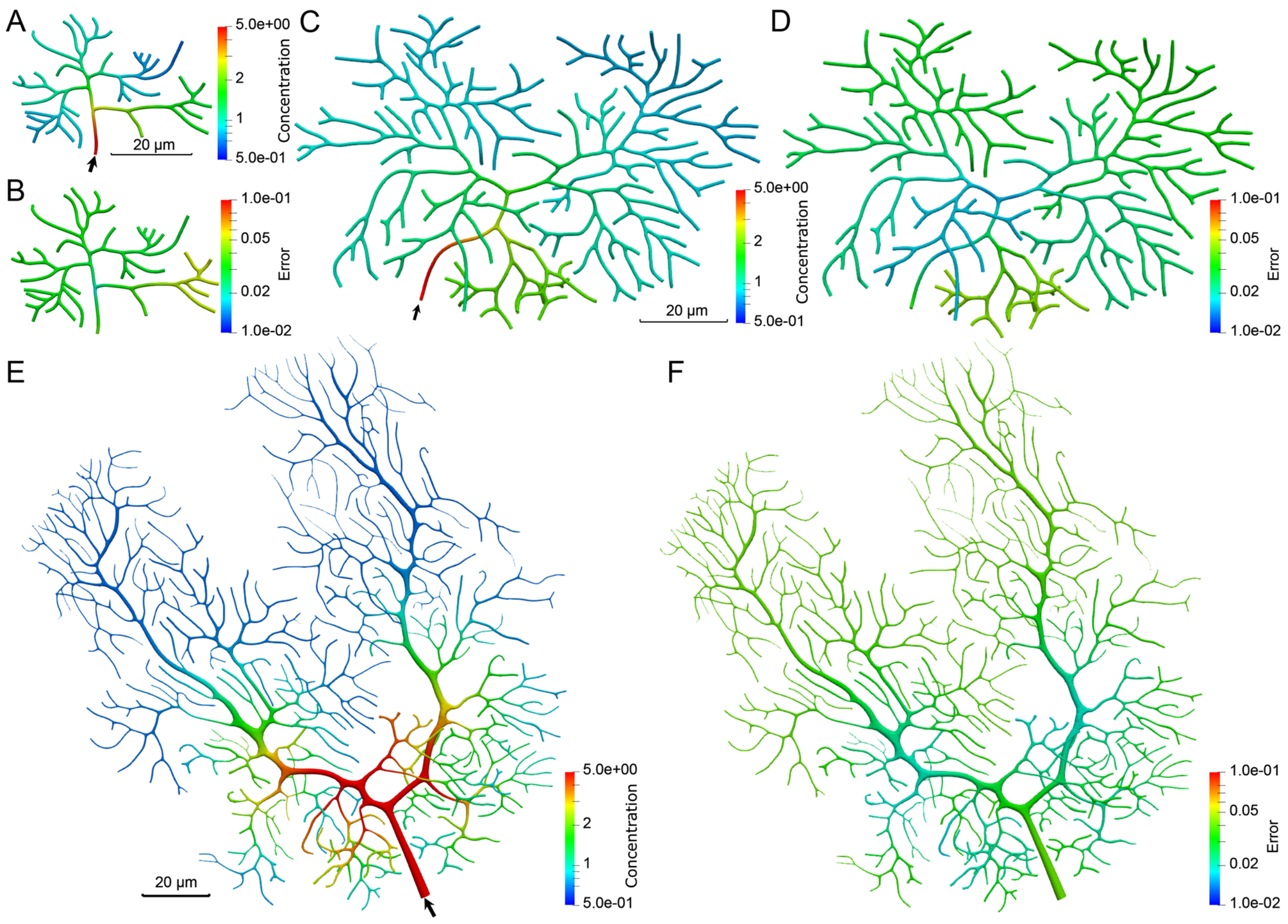

Figure 5. The concentration prediction of material transport in the neurite network geometry of NMO_54504, NMO_54499 and NMO_00865. (**A**, **C**, **E**) The predicted concentration results of steady state at $t = 12\ s, 41\ s$, and $63\ s$, respectively. The black arrows point to the inlet of material. Scalar bar: 20 μm. Unit for color bars: $mol/\mu m^3$. (**B**, **D**, **F**) The nodal errors between the ground truth and predicted concentration. Logarithmic scale is used to highlight the distribution pattern.

Species	Model name	Mesh (vertices, elements)	Bifurcation number	IGA computation (nodes, time (mins))	GNN prediction time (mins)	Speedup ratio (IGA vs GNN)	GNN prediction MRE (%)
Zebrafish	NMO_66731 (Fig. 3B)	(127,221, 112,500)	15	(8, 468)	1.6	293	6.7
	NMO_66748 (Fig. 3D)	(282,150, 249,660)	35	(10, 672)	3.9	172	7.3
	NMO_06846 (Fig. 4A)	(116,943, 101,880)	20	(8, 413)	2.1	197	7.2
	NMO_06840 (Fig. 4B)	(280,434, 248,040)	35	(10, 705)	4.4	160	7.5
Mouse	NMO_112145 (Fig. 4C)	(110,985, 98,460)	9	(8, 350)	1.2	291	7.4
	NMO_32235 (Fig. 4H)	(96,714, 85,680)	9	(6, 320)	1.3	246	7.8
	NMO_32280 (Fig. 4I)	(131,967, 117,000)	12	(8, 493)	1.5	329	8.1
	NMO_54504 (Fig. 5A)	(116,943, 101,880)	32	(8, 436)	3.1	140	8.3
	NMO_54499 (Fig. 5C)	(524,871 459,360)	127	(20, 759)	6.1	124	8.7
	NMO_00865 (Fig. 5E)	(1,350,864, 1,179,900)	356	(40, 908)	7.1	127	9.1

Table 2. Statistics of all tested complex neurite networks. Each node in Bridges has 28 cores.

to directly handle the IGA simulation data stored in the unstructured mesh. Given input boundary conditions and geometry information of a pipe or bifurcation, the well trained simulators are able to provide high accuracy results (MRE $< 7\%$). The second GNN-based assembly model then collects all the local predictions and follows the graph representation of the neurite network to update the global prediction. As shown in Figs. 4, 5, the model is evaluated on complex neurons from mouse and zebrafish and shows its applicability with consistent performance in different neuron geometries. In particular, the model is capable of providing the spatiotemporal concentration prediction with MRE $< 10\%$ and over 100 times faster than the IGA simulation. Furthermore, the accurate distribution prediction can help us determine high concentration regions, which is essential to infer possible locations to develop transport disruption.

Our study shows that the complex and diverse geometry of neurons has a major impact on the material concentration distribution and further affects the prediction accuracy of our GNN model. As shown in Figs. 4 and 5, the radius of most complex neurons is larger around the inlet region and decreases downstream, which contributes to material accumulation at the inlet region. The increase of bifurcations can also aggravate the accumulation when there is a sharp decrease of radius in the downstream branches (Fig. 5E). These geometry features contribute to the complicated concentration distribution and thus bring challenges to improve the performance of our GNN model. One challenge we have for the most complex neuron NMO_00865 is that MRE gets worse to 12.5% when it was initially tested using the model trained with the zebrafish neuron dataset. We plot the nodal error and find the error is higher at regions with high curvature or sharp radius change (Supplementary Fig. S5). The possible reason is that the GNN model lacks the knowledge of these scenarios since they are not common in the training dataset obtained from zebrafish neurons. To address this issue, we extract 20 geometries for each scenario from NMO_00865 and include them in the training dataset. We adopt the transfer learning method[44] to reuse the pre-trained GNN model as the starting point and obtain an improved model by training with the new dataset. The new model achieves the MRE of 9.1% on NMO_00865 which is comparable to the other testing neurons. This indicates that our GNN model can be further optimized with transfer learning on a better training dataset and an optimal dataset with a variety of geometries considered is quite essential to improve the applicability of the model.

The integration of the governing equations with the GNN model also plays a critical role in guiding the GNN model to learn the underlying physics behind the simulation data. By including the PDE residuals as the physics loss in the loss function of the GNN simulator (Eq. 1), superior performance is achieved over the model trained with the standard MSE loss function. The interface loss is also considered in the loss function of the GNN assembly model (Eq. 2) to minimize the noncontinuous concentration gap between the assembled geometries, which enables a continuous global concentration prediction. The results indicate that the physics-based loss function serves as an explainable component of the GNN model and teaches the model to utilize the simulation data more efficiently.

Our method, directly learning the transport mechanism from numerical simulation data in mesh format, makes it flexible to design data-driven DL models and further explore the value of simulation data. The model can also be extended to a general GNN framework for learning other PDE models in any complex geometry. Specifically, we can modify the PDE residuals in the simulator loss function to create a physics-guided data-driven DL model for any PDE solver. Our study also has its limitations, which we will address in our future work. In the current model, we only consider different geometries and boundary conditions as input features, while fixing the simulation parameters. In addition, there are still many more neurite networks with different topologies that are not considered in our dataset. To further improve the performance of our GNN model, we will optimize the training dataset by including different simulation parameter settings and different geometries to generalize its application in neurons of broader morphology. We will also study how to combine GNN with physics-informed neural network so that the geometry information of the data can be fully encoded and utilized in the DL framework. Despite these limitations, our GNN model efficiently and accurately predicts the dynamic concentration distribution within complex neurite networks and provides a powerful tool for further study in this field.

Methods

IGA-based material transport simulation on complex neuron geometries. In our previous study, we developed an IGA-based solver to perform dynamic material transport simulation in large and complex neurite networks[20]. IGA is an advanced finite element technique that differs from conventional finite element analysis (FEA) for its direct integration of geometrical modeling with numerical solutions. With the same smooth spline basis functions[45] utilized as the basis for both geometrical modeling and numerical solution, IGA can accurately represent a wide range of complex geometry with high-order continuity while offering superior performance over FEA in numerical accuracy and robustness[19]. Due to its many performance advantages, IGA has been adopted in a wide variety of research areas, such as linear elasticity[46], shell analysis[47–49], cardiovascular modeling[50–55] and fluid-structure interaction[56–58], as well as collocation[59, 60]. Truncated T-splines[61, 62] were developed to facilitate local refinement over unstructured quadrilateral and hexahedral meshes. Blended B-splines[63] and Catmull-Clark subdivision basis functions[64] were investigated to enable improved or even optimal convergence rates for IGA. Our geometric modeling and IGA simulation pipeline[20] is shown in Supplementary Fig. S4. To reconstruct the geometry of different neurite networks for the simulation, we first obtain their morphologies stored in the SWC format from the NeuroMorpho database[41]. With the skeleton and diameter information provided in the SWC files, we adopt the skeleton-sweeping method[50] to generate the hexahedral control mesh of the neurite network, and then build trivariate truncated hierarchical B-splines over it. Regarding the governing equations of the simulation, we generalize the 1D motor-assisted transport model[11] to 3D geometry to accurately account for the actual morphology of the neurite. The model is described as a group of "reaction–diffusion-transport" equations, we have

$$\begin{cases} \dfrac{\partial c_0}{\partial t} - D\nabla^2 c_0 = -(k_+ + k_-)c_0 + k'_+ c_+ + k'_- c_- & \text{in } \Omega, \\ \dfrac{\partial c_\pm}{\partial t} + \boldsymbol{u}_\pm \cdot \nabla c_\pm = k_\pm c_0 - k'_\pm c_\pm & \text{in } \Omega, \\ c_0 = c, c_+ = \lambda c & \text{at incoming end,} \\ c_0 = \tilde{c}, c_- = \tilde{\lambda}\tilde{c} & \text{at outgoing end,} \end{cases} \tag{3}$$

where the open set $\Omega \subset \mathbb{R}^3$ represents the internal space of the single neurite; c_0, c_+ and c_- are the spatial concentrations of free, incoming (relative to the cell body; retrograde), and outgoing (anterograde) materials, respectively; D is the diffusion coefficient of free materials; $\boldsymbol{u}_+$ and $\boldsymbol{u}_-$ are velocities of incoming and outgoing materials, respectively; $k_\pm$ and $k'_\pm$ are rates of cytoskeletal filament attachment and detachment of incoming and outgoing materials, respectively; and λ, $\tilde{\lambda}$ represent the degree of filament attachment at both ends and are also referred to as the "degree of loading"[11]. Regarding the boundary condition, we assume stable concentrations of free and incoming materials at both the incoming end and the outgoing end and set constant values to λ and $\tilde{\lambda}$. In this study, we assume the filament system is unipolar that leads to a unidirectional material transport process and ignore c_-, $\boldsymbol{u}_-$, k_-, k'_- terms in Eq. (3).

To obtain a physically realistic transport process in 3D, we assume that the flow of transport is incompressible and solve the steady-state Navier–Stokes equation to derive a physically realistic velocity field inside the single neurite

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 & \text{in } \Omega, \\ \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) + \nabla p = \nu \Delta \boldsymbol{u} + \boldsymbol{f} & \text{in } \Omega, \end{cases} \tag{4}$$

where the open set $\Omega \subset \mathbb{R}^3$ represents the incompressible fluid domain, $\boldsymbol{u}$ is the flow velocity, p is the pressure, $\boldsymbol{f}$ is the given body force per unit volume, ν is the kinematic viscosity, and $\otimes$ denotes the tensor product. Regarding boundary conditions, we impose non-slip condition at the neurite wall and apply a parabolic profile inlet velocity for each point on the circular cross section as

$$u(r) = u_i(1 - (r/R)^2), \tag{5}$$

where u_i is the inlet transport velocity defined in our material transport model, r is the distance from the center of the circular cross section to the point, and R is the radius of the circular cross section. The direction of the velocity is perpendicular to the inlet cross section. We utilize IGA to solve Eq. (4) and get the velocity field $\boldsymbol{u}$. Based on the unidirectional transport assumption, the velocity field is used as $\boldsymbol{u}_+ = \boldsymbol{u}$ to solve Eq. (3). As a result, we obtain the material concentration at every time step stored in the truncated hierarchical B-spline of the neurite network. These simulation results are then collected and processed as the training data for this study.

A review of GNNs.

GNN is a machine learning technique that was first proposed to generalize existing neural networks to operate on the graph domain[65]. It has been widely used to perform graph analysis in social networks[31, 66], traffic networks[67] and physics[68, 69]. In the following, we explain the basic idea of GNN by using the original GNN framework[65].

A graph $\mathscr{G}$ is a pair $(\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is the set of nodes and $\mathscr{E}$ is the set of edges. Given the node features $\mathbf{X}_\mathscr{V}$, the edge features $\mathbf{X}_\mathscr{E}$ and outputs $\mathbf{O}$, the GNN is trained to learn the embedding state $\mathbf{S}$ to establish the global mapping between all the features and outputs. Since each node is naturally defined by its features and the related nodes in a graph, the nodal embedding state $\mathbf{s}_v$ and the nodal output $\mathbf{o}_v$ can be produced locally as

$$\mathbf{s}_v = f_t(\mathbf{x}_v, \mathbf{x}_\epsilon, \mathbf{s}_{ne[v]}, \mathbf{x}_{ne[v]}), \tag{6}$$

$$\mathbf{o}_v = f_o(\mathbf{s}_v, \mathbf{x}_v), \tag{7}$$

where $\mathbf{x}_v, \mathbf{x}_e, \mathbf{s}_{ne[v]}, \mathbf{x}_{ne[v]}$ are the features of node v, the features of its edges, the embedding states and the features of nodes in the neighborhood of v, respectively. f_t is the *local transition function* and f_o is the *local output function*. Both f_t and f_o are parametric functions including the parameters to be trained and shared among all nodes.

Let F_t (the *global transition function*) and F_o (the *global output function*) be stacked versions of f_t and f_o for all nodes in a graph, respectively. Then, we get a compact form of the global mapping:

$$\mathbf{S} = F_t(\mathbf{S}, \mathbf{X}_\mathscr{V}, \mathbf{X}_\mathscr{E}), \tag{8}$$

$$\mathbf{O} = F_o(\mathbf{S}, \mathbf{X}_\mathscr{V}). \tag{9}$$

Based on Banach's fixed point theorem[70], GNN uses the following iterative scheme to compute the embedding states:

$$\mathbf{S}^{k+1} = F_t(\mathbf{S}^k, \mathbf{X}_\mathscr{V}, \mathbf{X}_\mathscr{E}), \tag{10}$$

where $\mathbf{S}^k$ is the k-th iteration of $\mathbf{S}$. The training of the aforementioned GNN model is straightforward. With the target output $\tilde{\mathbf{o}}_v$ for the supervision, the loss function can be defined as

$$loss = \sum_{i=1}^{p} ||\tilde{\mathbf{o}}_{v,i} - \mathbf{o}_{v,i}||, \tag{11}$$

where p is the number of supervised nodes. Then the gradient-descent strategy is used to update and learn the parameters in f_t and f_o.

The original GNN framework has the limitation that they can only handle the nodes embedded in fixed graphs and have to be re-trained whenever the graph structure changes. To address this limitation, GraphSAGE[31] was proposed to generalize the GNN algorithms to learn the nodes embedded in dynamic graphs. The key idea of GraphSAGE is to learn how to aggregate feature information from a node's local neighborhood. When the aggregator weights are learned, the embedding of an unseen node can be generated from its features and neighborhood. To further generalize the concept of GNN, several different GNN frameworks were proposed to integrate different DL models into one framework, such as the message passing neural network (MPNN)[71], the non-local neural network (NLNN)[72] and the graph network (GN)[36]. In our study, the extensive geometry of neurite networks contributes to different mesh structures for prediction. In particular, different lengths of the pipe lead to different numbers of cross sections along the pipe skeleton. Therefore, we implement GN in our GNN model to ensure the model is suitable for any neuron geometry.

Model evaluation. Two performance metrics were used to evaluate the accuracy of the predicted concentration distributions from our algorithm: mean absolute error (MAE) and mean relative error (MRE). For each predicted result, the MAE is defined by

$$MAE = \sqrt{\sum_{i=1}^{N} \frac{1}{N}(c_i^P - c_i^G)^2}, \tag{12}$$

where N denotes the number of elements in the output, c_i^P and c_i^G denote the predicted and ground truth concentration values of the i^{th} node in a given mesh, respectively. For each predicted result, the MRE is defined by

$$MRE = \frac{MAE}{\max||c^G|| - \min||c^G||} \times 100\%, \tag{13}$$

where $\max||c^G||$and $\min||c^G||$denote the maximum and minimum nodal concentration values from the ground truth result, respectively.

Data availability

All data generated during this study can be reconstructed by running the source code.

Code vailability

The source code for our model and all input data are available for download from a public software repository located at https://github.com/truthlive/NeuronTransportLearning.

Received: 10 January 2021; Accepted: 17 May 2021
Published online: 28 May 2021

References

1. Segev, I. & London, M. Untangling dendrites with quantitative models. *Science* **290**, 744–750 (2000).
2. Swanger, S. A. & Bassell, G. J. Dendritic protein synthesis in the normal and diseased brain. *Neuroscience* **232**, 106–127 (2013).
3. De Vos, K. J., Grierson, A. J., Ackerley, S. & Miller, C. C. Role of axonal transport in neurodegenerative diseases. *Nat. Rev. Neurosci.* **31**, 151–173 (2008).
4. Gunawardena, S. & Goldstein, L. S. Polyglutamine diseases and transport problems: deadly traffic jams on neuronal highways. *Arch. Neurol.* **62**, 46–51 (2005).
5. Millecamps, S. & Julien, J.-P. Axonal transport deficits and neurodegenerative diseases. *Nat. Rev. Neurosci.* **14**, 161–176 (2013).
6. Kononenko, N. L. *et al.* Retrograde transport of TrkB-containing autophagosomes via the adaptor AP-2 mediates neuronal complexity and prevents neurodegeneration. *Nat. Commun.* **8**, 1–16 (2017).
7. Zhang, H. *et al.* Modulation of AMPA receptor surface diffusion restores hippocampal plasticity and memory in Huntington's disease models. *Nat. Commun.* **9**, 1–16 (2018).
8. Vale, R. D. The molecular motor toolbox for intracellular transport. *Cell* **112**, 467–480 (2003).
9. Hirokawa, N., Niwa, S. & Tanaka, Y. Molecular motors in neurons: transport mechanisms and roles in brain function, development, and disease. *Neuron* **68**, 610–638 (2010).
10. Franker, M. A. & Hoogenraad, C. C. Microtubule-based transport-basic mechanisms, traffic rules and role in neurological pathogenesis. *J. Cell Sci.* **126**, 2319–2329 (2013).
11. Smith, D. & Simmons, R. Models of motor-assisted transport of intracellular particles. *Biophys. J.* **80**, 45–68 (2001).
12. Friedman, A. & Craciun, G. A model of intracellular transport of particles in an axon. *J. Math. Biol.* **51**, 217–246 (2005).
13. Craciun, G., Brown, A. & Friedman, A. A dynamical system model of neurofilament transport in axons. *J. Theor. Biol.* **237**, 316–322 (2005).
14. Kuznetsov, A. & Avramenko, A. A macroscopic model of traffic jams in axons. *Math. Biosci.* **218**, 142–152 (2009).
15. Kuznetsov, A., Avramenko, A. & Blinov, D. Modeling the effect of a microtubule swirl on fast axonal transport. *Int. Commun. Heat Mass Transf.* **37**, 234–238 (2010).
16. Hines, M. L. & Carnevale, N. T. The NEURON simulation environment. *Neural Comput.* **9**, 1179–1209 (1997).
17. Loew, L. M. & Schaff, J. C. The virtual cell: a software environment for computational cell biology. *Trends Biotechnol.* **19**, 401–406 (2001).

18. Joucla, S., Glière, A. & Yvert, B. Current approaches to model extracellular electrical neural microstimulation. *Front. Comput. Neurosci.* **8**, 13 (2014).
19. Hughes, T., Cottrell, J. & Bazilevs, Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Eng.* **194**, 4135–4195 (2005).
20. Li, A., Chai, X., Yang, G. & Zhang, Y. J. An isogeometric analysis computational platform for material transport simulation in complex neurite networks. *Mol. Cell. Biomech.* **16**, 123–140 (2019).
21. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **25**, 1097–1105 (2012).
22. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
23. Witten, I. H., Frank, E., Hall, M. A. & Pal, C. J. *Data Mining: Practical Machine Learning Tools and Techniques* (Morgan Kaufmann, 2016).
24. Han, J., Jentzen, A. & Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci.* **115**, 8505–8510 (2018).
25. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
26. He, S. *et al.* Learning to predict the cosmological structure formation. *Proc. Natl. Acad. Sci.* **116**, 13825–13832 (2019).
27. Farimani, A. B., Gomes, J. & Pande, V. S. Deep learning the physics of transport phenomena. arXiv:1709.02432 (2017).
28. Wiewel, S., Becher, M. & Thuerey, N. Latent space physics: towards learning the temporal evolution of fluid flow. *Comput. Graph. Forum* **38**, 71–82 (2019).
29. Li, A., Chen, R., Farimani, A. B. & Zhang, Y. J. Reaction diffusion system prediction based on convolutional neural network. *Sci. Rep.* **10**, 1–9 (2020).
30. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907 (2016).
31. Hamilton, W., Ying, Z. & Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems* 1024–1034 (2017).
32. Monti, F. *et al.* Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 5115–5124 (2017).
33. Fey, M., Lenssen, J. E., Weichert, F. & Müller, H. SplineCNN: fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 869–877 (2018).
34. Ogoke, F., Meidani, K., Hashemi, A. & Farimani, A. B. Graph convolutional neural networks for body force prediction. arXiv:2012.02232 (2020).
35. Karamad, M. *et al.* Orbital graph convolutional neural network for material property prediction. *Phys. Rev. Mater.* **4**, 093801 (2020).
36. Battaglia, P. W. *et al.* Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261 (2018).
37. Alet, F. *et al.* Graph element networks: adaptive, structured computation and memory. arXiv:1904.09019 (2019).
38. Sanchez-Gonzalez, A. *et al.* Learning to simulate complex physics with graph networks. arXiv:2002.09405 (2020).
39. Paszke, A. *et al.* Pytorch: an imperative style, high-performance deep learning library. In Wallach, H. *et al.* (eds.) *Advances in Neural Information Processing Systems*, Vol. 32, 8024–8035 (Curran Associates, Inc., 2019).
40. Fey, M. & Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. arXiv:1903.02428 (2019).
41. Ascoli, G. A., Donohue, D. E. & Halavi, M. Neuromorpho.org: a central resource for neuronal morphologies. *J. Neurosci.* **27**, 9247–9251 (2007).
42. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. arXiv:1412.6980 (2014).
43. Towns, J. *et al.* XSEDE: accelerating scientific discovery. *Comput. Sci. Eng.* **16**, 62–74 (2014).
44. Pan, S. J. & Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **22**, 1345–1359 (2009).
45. Piegl, L. & Tiller, W. *The NURBS Book* (Springer, 2012).
46. Dimitri, R. *et al.* Isogeometric large deformation frictionless contact using T-splines. *Comput. Methods Appl. Mech. Eng* **269**, 394–414 (2014).
47. Benson, D., Bazilevs, Y., Hsu, M.-C. & Hughes, T. J. Isogeometric shell analysis: the Reissner–Mindlin shell. *Comput. Methods Appl. Mech. Eng.* **199**, 276–289 (2010).
48. Casquero, H. *et al.* Seamless integration of design and Kirchhoff–Love shell analysis using analysis-suitable unstructured T-splines. *Comput. Methods Appl. Mech. Eng.* **360**, 112765 (2020).
49. Casquero, H. *et al.* Arbitrary-degree T-splines for isogeometric analysis of fully nonlinear Kirchhoff–Love shells. *Comput. Aided Des.* **82**, 140–153 (2017).
50. Zhang, Y., Bazilevs, Y., Goswami, S., Bajaj, C. L. & Hughes, T. J. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. *Comput. Methods Appl. Mech. Eng.* **196**, 2943–2959 (2007).
51. Zhang, Y. *et al.* An atlas-based geometry pipeline for cardiac Hermite model construction and diffusion tensor reorientation. *Med. Image Anal.* **16**, 1130–1141 (2012).
52. Zhang, Y. Challenges and advances in image-based geometric modeling and mesh generation. In Zhang, Y (ed.) *Image-Based Geometric Modeling and Mesh Generation* 1–10 (Springer, 2013).
53. Urick, B., Sanders, T. M., Hossain, S. S., Zhang, Y. J. & Hughes, T. J. Review of patient-specific vascular modeling: template-based isogeometric framework and the case for CAD. *Arch. Comput. Methods Eng.* **26**, 381–404 (2019).
54. Yu, Y., Zhang, Y. J., Takizawa, K., Tezduyar, T. E. & Sasaki, T. Anatomically realistic lumen motion representation in patient-specific space-time isogeometric flow analysis of coronary arteries with time-dependent medical-image data. *Comput. Mech.* **65**, 395–404 (2020).
55. Zhang, Y. J. *Geometric Modeling and Mesh Generation from Scanned Images* Vol. 6 (CRC Press, 2016).
56. Bazilevs, Y., Calo, V. M., Zhang, Y. & Hughes, T. J. Isogeometric fluid-structure interaction analysis with applications to arterial blood flow. *Comput. Mech.* **38**, 310–322 (2006).
57. Casquero, H., Liu, L., Bona-Casas, C., Zhang, Y. & Gomez, H. A hybrid variational-collocation immersed method for fluid-structure interaction using unstructured T-splines. *Int. J. Numer. Methods Eng.* **105**, 855–880 (2016).
58. Casquero, H., Zhang, Y. J., Bona-Casas, C., Dalcin, L. & Gomez, H. Non-body-fitted fluid-structure interaction: divergence-conforming B-splines, fully-implicit dynamics, and variational formulation. *J. Comput. Phys.* **374**, 625–653 (2018).
59. Anitescu, C., Jia, Y., Zhang, Y. J. & Rabczuk, T. An isogeometric collocation method using superconvergent points. *Comput. Methods Appl. Mecha. Eng.* **284**, 1073–1097 (2015).
60. Casquero, H., Liu, L., Zhang, Y., Reali, A. & Gomez, H. Isogeometric collocation using analysis-suitable T-splines of arbitrary degree. *Comput. Methods Appl. Mech. Eng.* **301**, 164–186 (2016).
61. Wei, X., Zhang, Y., Liu, L. & Hughes, T. J. Truncated T-splines: fundamentals and methods. *Comput. Methods Appl. Mech. Eng.* **316**, 349–372 (2017).
62. Wei, X., Zhang, Y. J. & Hughes, T. J. Truncated hierarchical tricubic C^0 spline construction on unstructured hexahedral meshes for isogeometric analysis applications. *Comput. Math. Appl.* **74**, 2203–2220 (2017).
63. Wei, X. *et al.* Blended B-spline construction on unstructured quadrilateral and hexahedral meshes with optimal convergence rates in isogeometric analysis. *Comput. Methods Appl. Mech. Eng.* **341**, 609–639 (2018).
64. Li, X., Wei, X. & Zhang, Y. J. Hybrid non-uniform recursive subdivision with improved convergence rates. *Comput. Methods Appl. Mech. Eng.* **352**, 606–624 (2019).

65. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **20**, 61–80 (2008).
66. Zhang, M. & Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems* 5165–5175 (2018).
67. Geng, X. *et al.* Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 3656–3663 (2019).
68. Battaglia, P. *et al.* Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems* 4502–4510 (2016).
69. Santoro, A. *et al.* A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems* 4967–4976 (2017).
70. Khamsi, M. A. & Kirk, W. A. *An Introduction to Metric Spaces and Fixed Point Theory* Vol. 53 (Wiley, 2011).
71. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70, 1263–1272 (2017).
72. Wang, X., Girshick, R., Gupta, A. & He, K. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 7794–7803 (2018).

Acknowledgements
The authors acknowledge the support of NSF Grants CMMI-1953323 and CBET-1804929. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the NSF Grant ACI-1548562. Specifically, it used the Bridges system, which is supported by the NSF Grant ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

Author contributions
A.L. designed the model and carried out the calculations and analysis with the instruction from Y.J.Z.. A.B.F. provided advice on the selection of machine learning methods. All authors contributed to the writing, discussions and revisions of the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-021-90724-3.

Correspondence and requests for materials should be addressed to Y.J.Z.