

MIT Open Access Articles

Rapid mechanical property prediction and <i>de novo</i> design of three-dimensional spider webs through graph and GraphPerceiver neural networks

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: Lu, Wei, Yang, Zhenze and Buehler, Markus J. 2022. "Rapid mechanical property prediction and <i>de novo</i> design of three-dimensional spider webs through graph and GraphPerceiver neural networks." 132 [7].

As Published: 10.1063/5.0097589

Publisher: AIP Publishing

Persistent URL: <https://hdl.handle.net/1721.1/145503>

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Terms of use: Creative Commons Attribution 4.0 International license



Rapid mechanical property prediction and *de novo* design of three-dimensional spider webs through graph and GraphPerceiver neural networks

F SCI

Cite as: J. Appl. Phys. **132**, 074703 (2022); <https://doi.org/10.1063/5.0097589>

Submitted: 29 April 2022 • Accepted: 12 July 2022 • Published Online: 17 August 2022

Published open access through an agreement with Massachusetts Institute of Technology

id Wei Lu, id Zhenze Yang and id Markus J. Buehler

COLLECTIONS

Paper published as part of the special topic on [Advances in Multi-Scale Mechanical Characterization](#)

F This paper was selected as Featured

SCI This paper was selected as Scilight



View Online



Export Citation



CrossMark

ARTICLES YOU MAY BE INTERESTED IN

[Computational method to create synthetic 3D spider web structures](#)Scilight **2022**, 341103 (2022); <https://doi.org/10.1063/10.0013749>[Plasmas for in situ resource utilization on Mars: Fuels, life support, and agriculture](#)Journal of Applied Physics **132**, 070902 (2022); <https://doi.org/10.1063/5.0098011>[Artificial photosynthetic monolithic devices using voltage-matched perovskite/silicon tandem photovoltaic modules](#)Journal of Applied Physics **132**, 075002 (2022); <https://doi.org/10.1063/5.0097485>

Trailblazers. New

Meet the Lock-in Amplifiers that measure microwaves.

Zurich Instruments [Find out more](#)

Rapid mechanical property prediction and *de novo* design of three-dimensional spider webs through graph and GraphPerceiver neural networks



Cite as: J. Appl. Phys. **132**, 074703 (2022); doi: [10.1063/5.0097589](https://doi.org/10.1063/5.0097589)

Submitted: 29 April 2022 · Accepted: 12 July 2022 ·

Published Online: 17 August 2022



Wei Lu,^{1,2} , Zhenze Yang,^{1,3} and Markus J. Buehler^{1,2,4,a)}

AFFILIATIONS

¹Laboratory for Atomistic and Molecular Mechanics (LAMM), Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, Massachusetts 02139, USA

²Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, Massachusetts 02139, USA

³Department of Materials Science and Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, Massachusetts 02139, USA

⁴Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, Massachusetts 02139, USA

Note: This paper is part of the Special Topic on Advances in Multi-Scale Mechanical Characterization.

a) Author to whom correspondence should be addressed: mbuehler@MIT.EDU. Tel.: +1-617-452-2750.

ABSTRACT

Spider webs feature advanced structural performance due to the evolutionary success of over more than 3×10^9 years, including lightweight design and exceptional mechanical properties. Spider webs are appealing for bio-inspired design since web designs serve multiple functions including mechanical protection and prey catching. However, high computational cost and limited quantified web properties render extensive spider web studies challenging in part due to the high structural complexity and randomness of fiber arrangements in 3D webs. Here, we report a computational method to relate spider web graph microstructures to effective mechanical properties, focusing on strength and toughness, and upscaling from the microscopic to the mesoscale level. The new computational framework uses deep neural networks, trained on graph-structured *Cyrtophora citricola* spider web mechanical data, in order to capture complex cross-scale structural relationships. Three different models are developed and compared. First, two Graph Neural Network (GNN) models, a Graph Convolutional Network, and a Principal Neighborhood Aggregation method. Second, a GraphPerceiver transformer model that is fed similar input data as provided to the GNN approach but within a natural language modeling context using self-attention mechanisms. The GraphPerceiver model can achieve similar performance as the GNN model, offering added flexibility for building deep learning models of diverse hierarchical biological materials. As an application of the model, we propose a computational optimization tool for synthetic web design that is used to generate synthetic, *de novo* spider web architectures. Finally, multi-objective optimization enables us to discover web structures that meet specific mechanical properties as design objectives.

© 2022 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>). <https://doi.org/10.1063/5.0097589>

I. INTRODUCTION

Spiders represent abundant species that have been around for over 3×10^9 years, owing in large part due to their evolutionary success. Spider webs with their unique hierarchical structures, exceptional mechanical properties, appealing architectural features, and multiple functions, have inspired studies in multiple areas,

including biomaterial,^{1–3} structure,^{4–6} and art.^{7–9} Specifically, spider webs are lightweight but display high structural performance, such as strength, toughness, and elasticity.^{4,6,10} They have varying geometric shapes depending on the types of webs¹¹ (e.g., orb, sheet, triangular, tube, etc.), and even a single web contains local variations with distinct mechanical properties.¹² The multiple-purpose web architecture

is designed for food catching, protection, and movements and is constructed along with a coordinated repair process.⁴

However, a gap exists between experimental and computational studies for spider webs. High computational cost is necessary to carry out dynamic simulations, and limited quantified web properties are available due to its structural complexity and high degree of randomness of silk arrangement. Thus, developing an effective tool incorporating advanced modeling techniques is critical for spider web studies, and the modeling of similarly complex architecture biological materials to understand structure–property relationships and to explore the vast architectural spaces. In recent years, artificial neural networks have emerged as powerful tools in various research areas since underlying data features could be efficiently learned through deep learning.¹³ Examples include but are not limited to, for instance, composite design,^{14,15} fracture prediction,¹⁶ and material design from sound and music,¹⁷ and many others where complex cross-domain transitions are necessary.

Graph neural networks (GNNs) represent a machine learning algorithm used as an optimizable information transformation with non-Euclidean structured data as input, which contains not only graph entity representations but also relationships among all the components.¹⁸ Whereas, similar functions could be achieved through the attention-based transformer neural networks, where the attention mechanism enables accessible weight distribution of factors for input data, and the transformer allows data form conversion.¹⁹ GNNs have found multiple applications in diverse works, including computer vision,^{20–22} natural language processing,^{23–26} social network recommendation,^{27,28} object detection,^{29–33} traffic forecasting and path search,^{34–36} drug discovery and interaction,^{37–41} and molecular prediction.^{42–44} Additionally, applications have been developed in the area of structural engineering, for example, the semi-supervised approach for architected material design,⁴⁵ and graph-based surrogate models for nodal displacement prediction of trusses.⁴⁶

In this paper, we develop and apply a computational framework for fast prediction of mechanical properties targeting strength and toughness for spider web structures using GNNs and transformer models. Based on the GNN architecture, we examine the use of both graph convolutional network (GCN)¹⁸ and principal neighborhood aggregation (PNA).⁴⁷ The developed prediction models are further used for synthetic web design and optimization. This method combines numerical modeling advances and practical analysis and is useful for spider web inspired design applications, such as *de novo* web architecture innovation and structure optimization. Compared with images, graph data are more instructive and flexible for spider web (and other similar architected biological) structures and represent a natural way to describe such systems. Moreover, graph data capture the specific structural relationships among all nodes and silks and correspondingly interpret the underlying mechanical features. Yet, challenges are present in operating graph-structured data due to the inconsistent and complex distribution of nodes, dependencies among edges, and larger data size,⁴⁸ this method of implementing GNN has several key contributions:

1. GNN implementation lowers computational cost for property prediction, compared with conventional numerical simulation such as structural finite element analysis methods. The computation process is simplified since the prediction using GNN is a

repeated operation once the model is constructed, for models of any size (especially important for novel web designs). Owing to its generalization capacity, fewer modeling parameters are required for property captures, such as boundary conditions and material properties.

2. Structural features of spider webs, which are intrinsically graphs, can be well learned using GNN models, including intricate silk arrangement and local variations since the models operate on graph data, which contains web geometries and their entity relationships.
3. A property prediction method could be implemented for computational design for synthetic spider webs for design space exploration and inspiration.
4. The developed methods can be used as a basis for rapid digital synthesis and analysis of spider web structures with the potential to be applied as validation and optimization tools for structural design.
5. The method developed here for providing graph data to the GNN model can alternatively be used to build a high-dimensional transformer model, realized in the GraphPerceiver architecture, offering an alternative way to use graph data, and providing even greater flexibility for bio-inspired applications to capture the mechanics of architected bio-structures.

This paper is organized as follows. First, the spider web data preparation methods are introduced along with the applied GNN and transformer structures, model development, and training processes. Then, the performances of different models are analyzed and compared with error and sensitive analysis implemented for model diagnosis and performance improvement. Afterward, an implementation of the prediction model is present for synthetic web generation and optimization. Lastly, a summary is concluded with potential impacts and future works discussed.

II. DATA SET, MODELS, AND METHODS

Sec. II presents the spider web data source, data preparation including the customization and implementations, the applied GNN algorithms and corresponding developed network architectures, as well as the computational framework of the proposed property prediction method.

A. Spider web data

The digital spider web data set was reported in Su *et al.*¹⁰ with spider webs being constructed, laser scanned, and digitally processed to transfer images to validated graph data; and mechanical web properties were obtained as reported in Ref. 49. The physical web was built by a *Cyrtophora citricola* spider [Fig. 1(a)]. The web construction process was carried out in a dim environment which is preferred by spiders, and the web was placed inside a rectangular carbon fiber frame above water trays to prevent spiders from escaping.¹⁰ With a physical web developed, sliced 2D images were captured using a sheet laser and a high-resolution EOS-1D X Canon camera with the capturing speed precisely controlled.¹⁰ An automatic remote scanning method developed by Su *et al.*,¹⁰ is implemented with reduced light interference from human activities. Afterward, the digital 3D web model is assembled from 2D

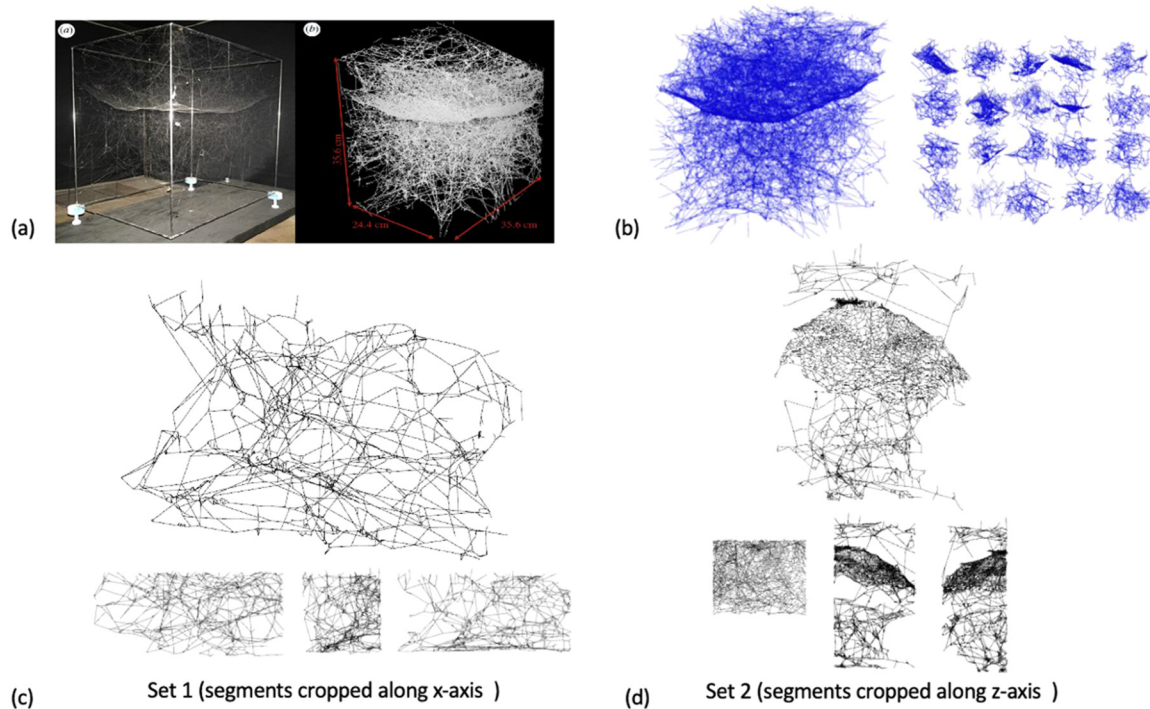


FIG. 1. Spider web data sets used in this study. (a) Image of the laboratory *Cyrtophora Citricola* spider web, and laser scanned web image generated as reported in Ref. 49. (b) Full-size spider webs structure after image processing and its smaller cubic samples processed by Ref. 10. (c) and (d) The two sets of spider web segments used as graph-structured data input for the regression model (set 1 and set 2, shown in perspective, top, front, and side views, respectively), developed by Ref. 12 using an innovative web graph augmentation method, visualized using grasshopper. These two segments are captured along the x-axis and z-axis, respectively.

images using an image processing algorithm as proposed in Ref. 10, including noise reduction for pixel filtering, dilation for constructing 3D structures, skeleton for adjusting edge width, and branch filtering to clean up networks. Mechanical properties of the spider web segments are simulated with mesoscale bead-spring models through a molecular dynamics program, Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS).^{50,51} Uniaxial stretching is applied for simulations of the strengths and toughness of web segments in x , y , and z directions, where stress is the ultimate stress indicated in the stress-strain curve and toughness is measured as the area under the curve.⁴⁹

With web segments taken from the full spider web [Fig. 1(b)] and their mechanical properties obtained, a data augmentation method is applied to expand the data set for computation, as elaborated in Ref. 12. Two pairs of neighboring web samples connected along the x - and z -axes, respectively, are selected [Figs. 1(c) and 1(d)]. New web segments are then generated by slicing the connected web sections along corresponding orthogonal axes with different combining ratios (100 sets of ratios per web sample), and their mechanical properties are interpolated with the same ratio combinations. As a result, a total of 200 unique spider web segments with corresponding mechanical properties are generated. This set forms the basis for neural network training of the three models developed in this study.

B. Data preparation

The digital spider web data are visualized using either *Grasshopper* through *Rhino3D*, or *Plotly*, a visualization package through python. Web geometries are transferred into graph-structured data using PyTorch Geometric (PyG, a library for graph neural network computation in Python)⁵² and then loaded into the GNN and transformer models for training. The previously generated spider web data are collected in comma-separated values (CSV) files, with nodes indicated with three-dimensional Cartesian coordinates, edge connectivity with the index of connecting nodes at two ends of the silks, and each graph with a specific property value for average strength or toughness.

First, for web data visualization: (1) With *Grasshopper*, the CSV files are first imported, then nodes and edges are plotted using coordination and node index, respectively. The combined geometries are baked into *Rhino* for visualization. (2) Using *go.Scatter3d* in *Plotly*, the nodes are plotted with markers with three-dimensional coordinates, and edges are plotted as discontinuous lines with every first two points presenting the two connecting nodes for every edge, and every third point with values of “None” for disconnection.

Second, as for data processing, all imported data are treated as tensors in PyG. As the web graph example shown in Fig. 2, for

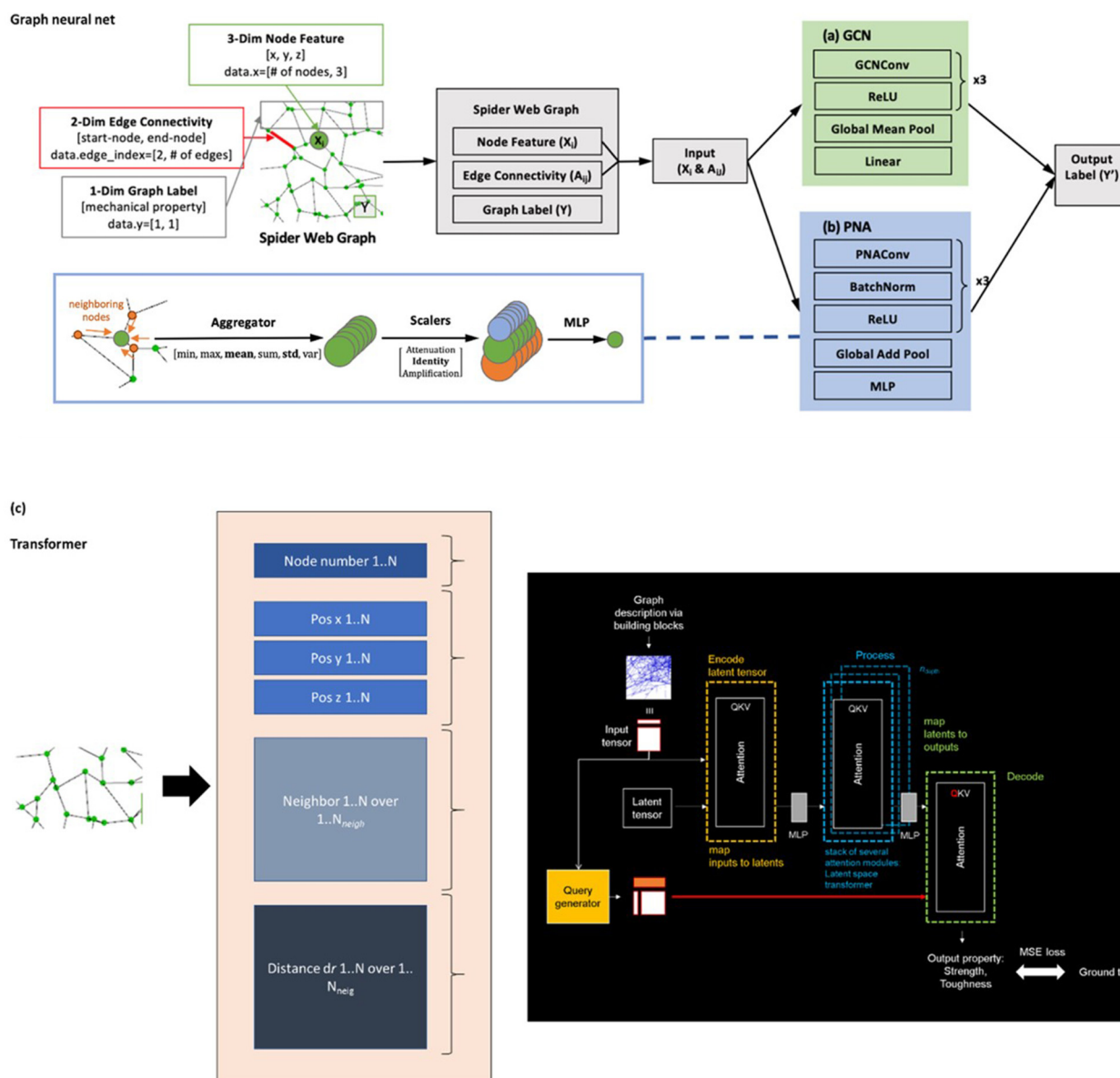


FIG. 2. Spider web graph and prediction model structures of (a) graph convolutional network (GCN) and (b) principal neighborhood aggregation (PNA), and (c) the GraphPerceiver transformer model to describe graph structures through embeddings. The node feature (X_i) and edge connectivity (adjacency matrix A_{ij}) are input to the graph convolutional layers. Message passing is repeated in each layer, neighborhood information is obtained, node embedding is aggregated to the next layer, graph embedding is trained, and graph label (Y') is predicted and optimized by comparing actual graph property (Y). Both (a) GCN and (b) PNA model have three convolutional layers, while (b) the PNA model has additional BatchNorm layers between every hidden layer to normalize the output from mini-batches to stabilize its learning process. In addition, the PNA structure has more aggregators and scalars, allowing continuous feature learning from neighborhoods. Panel (c) shows details of the transformer model, the GraphPerceiver. Graph specific information is provided via high-dimensional embeddings, which are then processed in a deep attention-model, resulting in prediction of graph-level properties.

each graph data, the node coordinates are loaded as three-dimensional node features, edge features are edge lengths (not applied in model constructions but have their impact analyzed in Sec. III C), the two connecting nodes for edges are customized as

two-dimensional edge connectivity, and the web property is loaded as a one-dimensional graph label. The strength and toughness of each web segment are defined as the average values of the corresponding properties of all three directions (x , y , and z). The web

strength and toughness are obtained for each of the models and properly labeled for later comparisons and validation.

As indicated in Sec. II A, there is a total of 200 data available for this study with 100 web segments from each spider web set. These two sets of spider webs have varied structural geometries. Web set 1 [Fig. 1(c)] contains more evenly arranged nodes although silk density varies locally, while web set 2 [Fig. 1(d)] comprises a distinct tent web section with high silk densities and looser edge entanglements. The feature distribution and property relationships of the original data set are shown in Fig. S1 in the [supplementary material](#), where nearly normal distributions of strength and toughness are present and an overall positive linear relationship between these two properties. This data set is implemented in two ways for training of the GNN models.

1. A single web set (100 graph data) is used for both model training and value prediction, with data randomly split with a ratio of 6:2:2 for training, validation, and testing.
2. A mixed set of segments from both set 1 and set 2 (200 graph data) are assembled, with the same splitting ratio (6:2:2) applied for training, validation, and testing sets.

For data operations, the mean absolute error (MAE) is applied for error and accuracy operation, and coefficient of determination (R2) for correlation of statistics between the predicted value and ground truth. However, it is noted the current data set has limitations. The data set is still rather small with a total of 200 web graphs, and the web graphs are derived from the same full spider web using the augmentation method, as illustrated in Sec. II A.

C. Graph neural network (GNN) development

The GNN models are developed using an open-source deep-learning library, PyTorch,⁵³ and its extended graph neural network tool package, PyG,⁵² in Python. Two GNN architectures, GCN and PNA, are implemented. GCN is one of the most commonly used network structures, which introduces the convolution mechanism into GNN to better extract local features for global prediction for non-Euclidean data with random connectivity.¹⁸ GCN operates spectral graph input and hidden features using a layer-wise propagation rule based on graph theory,¹⁸ as indicated in Eq. (1),

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right), \quad (1)$$

where $H^{(l)}$ is the matrix of hidden representation in the l th layer, with $H^{(0)} = X$ as the feature representation in the input layer. $W^{(l)}$ denotes the trainable weight matrix in the l th layer, and σ represents the activation function. $\tilde{A} = A + I$ donates the self-looped adjacency matrix with identity feature included, where A is the original adjacency matrix of the undirected graph and I is the identity matrix. \tilde{D} is the degree matrix of \tilde{A} , as known as the degree matrix of A with self-connection. \tilde{A} is normalized through $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$.¹⁸

The PNA structure is implemented for comparison since it has been shown to display enhanced performance for both node and graph-level tasks, especially graph regression.⁴⁷ Compared with GCN convolutional layers which applies a single aggregator

(mean or sum), PNA is more powerful specifically for feature learning since it manipulates a combined set of various aggregators (min, max, mean, sum, std, and var) and scalers (identity, attenuation, and amplification), which can retain continuous features from the neighborhoods and indicates a higher learning ability.⁴⁷ Specifically, aggregators define the statistic extraction methods for node features, and mixed operators can better distinguish the neighborhood messages. Scalers identify how messages are being aggregated, and a collective utilization generalizes the signal operation based on the degree of nodes.⁴⁷ Moreover, as may be relevant for future applications, PNA allows directed graphs as input, which could lower computational cost, and is useful if asymmetrical relationships are present in networks. PNA's message passing with combined aggregation for spatial features is indicated in Eq. (2),⁴⁷

$$X_i^{(t+1)} = U\left(X_i^{(t)}, \bigoplus_{(j,i) \in E} M(X_i^{(t)}, E_{j \rightarrow i}, X_j^{(t)})\right), \quad (2)$$

where $X_i^{(t)}$ and $X_j^{(t)}$ denote the features of node i and j , respectively. $E_{j \rightarrow i}$ is the feature of edge (j, i) in the directed graph and could be substituted as E_{ij} if the graph is undirected, while no edge features are implemented in this study. M and U are multilayer perceptron (MLPs), and U retrieves the feature dimension of hidden layers after the message is concatenated. $\bigoplus_{(j,i) \in E}$ denotes the PNA operator

with combined aggregators and scalers, for node j belongs to the neighborhood of node i .⁴⁷

An overview of the neural network models used for this work is described in Fig. 2. For both graph neural network architectures [Figs. 2(a) and 2(b)], three convolutional layers are implemented with message passing repeated across them. Information about each node is obtained from its first-order neighborhoods based on the connectivity, and node embeddings are aggregated to the next layer. The model is trained on the graph label, and the optimization is operated by comparing the model output with ground truth. For GCN models, every convolutional layer is followed by the activation function, Rectified Linear Unit (ReLU), for non-linear mapping to higher dimension space with an input dimension of 3 (equal to the dimension of input node features), hidden layer dimension of 32, and output size of 1 (equal to the dimension of graph label), as illustrated in Fig. S2 in the [supplementary material](#).

Graph-level labels are obtained from averaging the node embeddings using global mean pooling and mapping through a linear transformation. Moreover, the PNA model has additional BatchNorm layers between every hidden layer to stabilize the learning process by normalizing the outputs from batches. Global add pooling is applied in the PNA model with graph embeddings being collected through addition operation. The prediction is returned from a three-layer MLP. A sequential block with three input layer dimensions of 45, 30, and 15, activated by a non-linear ReLU function, and output with 1 neuron (Fig. S2 in the [supplementary material](#)). Moreover, all three scalers (identity, attenuation, and amplification) and two out of six aggregators (mean and standard deviation) are applied in the PNA model. This PNA model structure is more complex than the GCN model and has a higher number of model parameters, which is around 35 times larger, as summarized in Fig. S2 in the [supplementary material](#).

D. Training GNN model and inference

The general computational framework is indicated in Fig. 3, with the graph data developed and the graph properties simulated (as indicated in Sec. II B), the data set is customized with node features, edge connectivity, and graph label. Data are split into training, validation, and testing data sets with a ratio of 6:2:2 and loaded in batches into GNNs for supervised learning. The best trained model is selected and used to develop the inference model for corresponding web property prediction, including strength and toughness. The models are operated either on a local station with a NVIDIA GeForce RTX 2080 GPU with 31 GB memory or in a cluster with a NVIDIA Tesla V100 GPU and 16 GB memory each in a larger-scale cluster. The resulting inference (prediction) model is implemented for computational optimization for synthetic web generation.

As for the GNN training process, customized data are loaded into mini-batches with a batch size of 16. The size is selected and adjusted according to data set size, memory requirement, and potential overfitting issues. The loss is computed as the MAE between the output value and the actual graph property, and the accuracy is calculated as the average relative error every iteration. The GCN and PNA models are trained for 500 epochs. The training loss is optimized using *Adam* optimizer,⁵⁴ and the scheduler, *ReduceLROnPlateau*, with an initial rate of 10^{-3} and a minimum of 10^{-5} , is applied to reduce the learning rate by a factor of 0.5, when validation loss is not improved after ten epochs. A better model will be saved if higher testing accuracy is attained throughout the iterations. The hyperparameters of each model are selected and modified for better prediction performance, as indicated in Sec. III C.

As for inference model construction, the best model saved from the training process is imported into corresponding models. The configuration and structure of the neural network, as well as

the hyperparameters for inference models are required to be the same as the imported trained model. Then web properties can be predicted and returned with customized spider web geometries as input. As mentioned in Sec. II A, the original spider web properties are simulated through uniaxially stretching as mesoscale bead-spring particles.¹² We further note that the entire process is transferable for data simulated using different boundary conditions. For example, shear deformation could be applied through particle simulations to obtain the shear strength and shear modulus of spider web structures, and the same procedures could be applied for the constructed machine learning model.

E. GraphPerceiver model: Design and training

To compare against an alternative and distinct neural network architecture, we use an attention-based description of a graph model, as summarized in Fig. 2(c). The model uses a computationally efficient transformer implementation as proposed in Ref. 55, which can be scaled easily to very high-dimensional input and output data. The graph structure is described as summarized in the left side of Fig. 2(c), where node numbers, positions, neighbors, and distances to neighbors are provided to the transformer model via high-dimensional embeddings. A maximum number of neighbors is set; and chosen to be 5 for the web structures analyzed here (the number is determined by analyzing the dataset and can easily be chosen to meet the maximum expected number of neighbors in a system, and there exists no fundamental limit on the size of the neighbor list). The structural information about node number and neighbors and all other structural features are embedded as described in Table I.

We use *MSE* loss, an *Adam* optimizer with 0.000 01 learning rate, and the parameter betas = (0.99, 0.999).⁵⁴ Table I provides all the other hyperparameters used. The strength/toughness values are scaled to be within 0–1, and the positions and distances are scaled

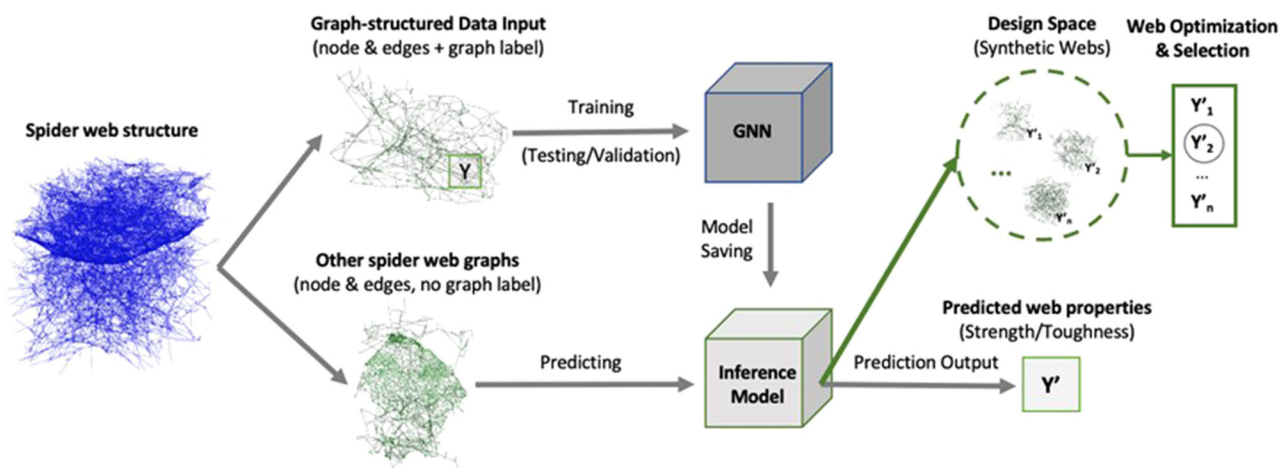


FIG. 3. Schematic of training and prediction processes for mechanical properties of spider web structures through graph neural network (GNN). With the spider web structures obtained through laser scanning, image processing, and segmentation, the web data set is loaded as graph-structured data with a computed mechanical property as the graph label. Data set is split for training, testing, and validation through GNN with a ratio of 6:2:2. The trained model is used in the inference model for corresponding web mechanical property prediction and is later on implemented for synthetic, *de novo* web selection and optimization.

TABLE I. Details of the GraphPerceiver model based on a transformer architecture for the strength and toughness models. For both cases, we use MSE loss, an Adam optimizer with 0.000 01 learning rate, and betas = (0.99, 0.999).

Feature	Strength	Toughness
Embedding node number	6	6
Embedding neighbor node numbers	6	6
Node position dimension	3	3
Neighbor distance dimension	1	1
Latent dim	768	512
Number of latents	768	512
Depth	10	10
Cross-attention heads	1	1
Latent heads	64	64
Cross_dim heads	128	128
Latent_dim heads	128	128
Weight-tie layers	False	False
Attn/FF dropout	0.2	0.2

by calculating the mean value and standard deviation for a variable. For each value, we then subtract the mean value and divide it by the standard deviation.

III. RESULTS AND DISCUSSION

In this section, the prediction method for spider web properties (workflow see Sec. II D) is processed with the customized web data set (discussed in Sec. II B). The training and prediction performance of applying both GCN and PNA architectures are presented along with the transformer model for comparison. Several model hyperparameters and optimization techniques are examined for the developed model structure to better understand how they influence model performance. A prediction model is implemented and then used for synthetic web design in Sec. IV, as a way to demonstrate the usefulness of the proposed model for structural applications in the context of design of novel web structures optimized for certain mechanical objectives.

A. GCN and PNA model performance

The training and prediction performances of GCN and PNA models are demonstrated in Figs. 4 and 5, respectively. In both figures, plots (1) and (2) show the performance of strength property for the single set and the mixed set of web segments, while (3) and (4) display the performance of toughness for two spider web data sets. These plots (from left to right) indicate the loss history in logarithmic values and accuracy for all training, validation, and testing data set, also the correlations of the prediction values and ground truth.

As shown in Figs. 4(1) and 4(2), the training process of the GCN model shows decent performance for spider web strength prediction with a testing accuracy of over 93% for both single and mixed data set input. Relatively, low loss and good convergence are seen in all training, validation, and testing data sets although slight overfitting exists since the training set has a lower loss value than validation and testing sets after convergence, whereas the training

accuracy of the mixed data set slightly surpasses the single one and displays a more effective learning process with a smoother loss history (this is likely due to the fact that a more generalized learning is processed with more diverse data seen by the model at every batch). The web strength prediction using the GCN model further shows sound correlations with ground truth data with R2 of 0.75 and 0.79 for single and mixed set predictions. In addition, the toughness model performs well [Figs. 4(3) and 4(4)] with accuracy values of around 95.7% and 93.2% for the single and mixed data sets though the correlations show more variation (R2 equals 0.86 and 0.64) than strength prediction.

Compared with GCN models, the PNA model shows less stable behavior in strength prediction but outperforms in toughness prediction. As shown in Figs. 5(1) and 5(2), even though PNA models show smoother training convergence in general, relatively larger and continuous loss variations present throughout the whole training process. For the single and mixed web sets, the strength accuracies are about 94.4% and 93.4% in training models with a correlation of around 0.87 and 0.56. The variation may result from the incongruity between the small-sized data set and the more complex neural network structure. In addition, the mixed data set displays slightly worse performance than the single set since overfitting appears to be magnified with more distinct web features present in the limited data set. However, in general, PNA models perform better in toughness prediction than GCN [Figs. 5(3) and 5(4)] with an accuracy of 97.3% and 95.6% and R2 of 0.90 and 0.84, respectively, which implies the capability of the PNA model in obtaining overall graph features through convolutional layers since toughness, denotes as the ability of energy absorption during the entire stretching process before rupture, is regarded as a more global mechanical feature than strength.

B. Error analysis of graph neural networks

As the results shown in Sec. III A indicated, the PNA model shows higher loss variation than the GCN model, at least based on the current spider web data set and network structure implemented. Our speculation is that the incompatibility between the small data set and the more complex PNA model. The PNA model is more complex in its formulation than the GCN model due to its combined aggregation operator with more aggregators and scalars applied, and the higher number of model parameters it, therefore, has (Fig. S2 in the supplementary material). As shown in Fig. S3(a) in the supplementary material, the PNA model has an overall lower training loss and higher testing accuracy than the GCN model, but larger loss variation among the epochs, which indicates the PNA model can return a good fit for a given batch but restricted performance through batches among the small data set. From another perspective, when the random state (an initialization parameter for random generator⁵⁶) is changed for data set splitting, the disparity in testing accuracy exists [Fig. S3(b) in the supplementary material]. Different sets of testing data show various responses to the model training, which conversely demonstrate the overfitting issue. Nevertheless, this speculation cannot yet be generalized since the data set we had is still small but will take data augmentation as a future step; besides, the property prediction method can be further improved by

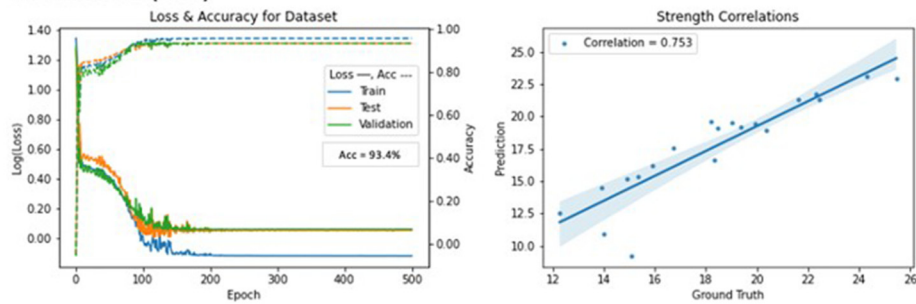
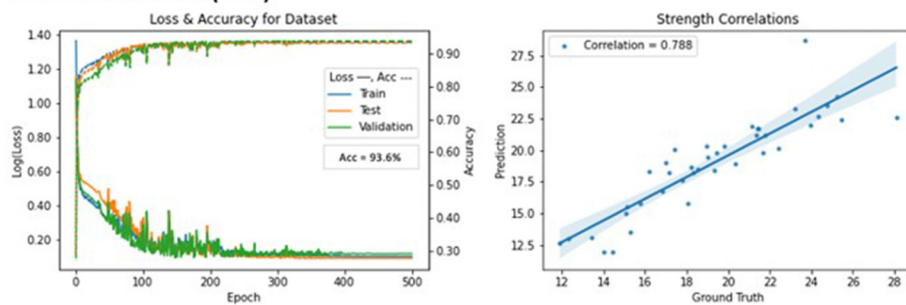
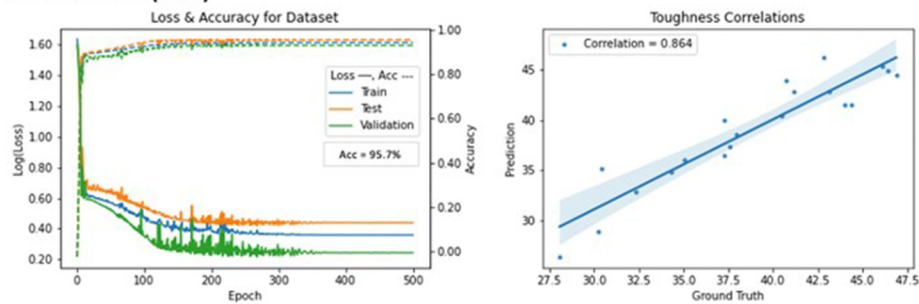
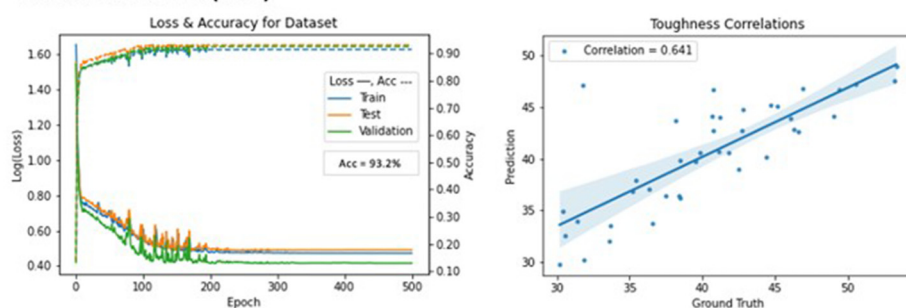
STRENGTH:**1. Full SET1 (100)****2. Full Mixed Set (200)****TOUGHNESS****3. Full SET1 (100)****4. Full Mixed Set (200)**

FIG. 4. Training and property prediction performance of the GCN model. (1) and (2) show the strength prediction for the single set of spider web segments and the mixed set of web segments, while (3) and (4) show training and predicting performance regarding spider web toughness. The plots (from left to right) are the loss in logarithmic value, and accuracy for training, testing, and validation data sets, with test accuracy labeled; the correlation between prediction value and ground truth, as well as the coefficient of determination (R^2) labeled. Loss and accuracy are calculated using MAE.

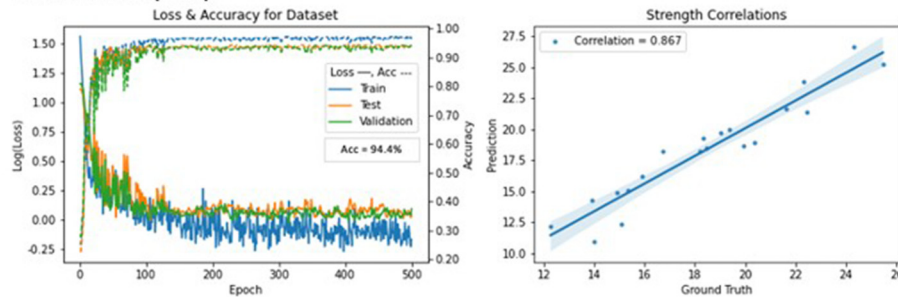
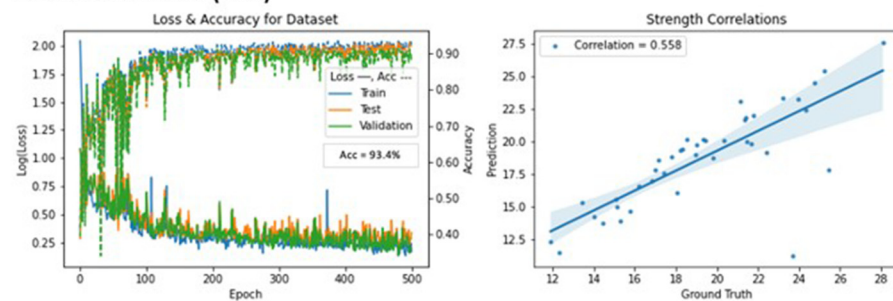
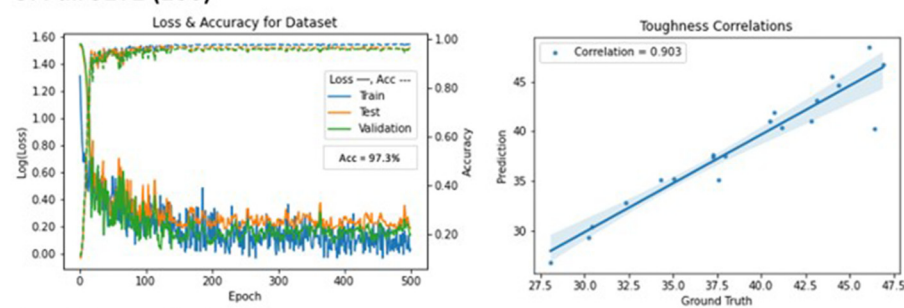
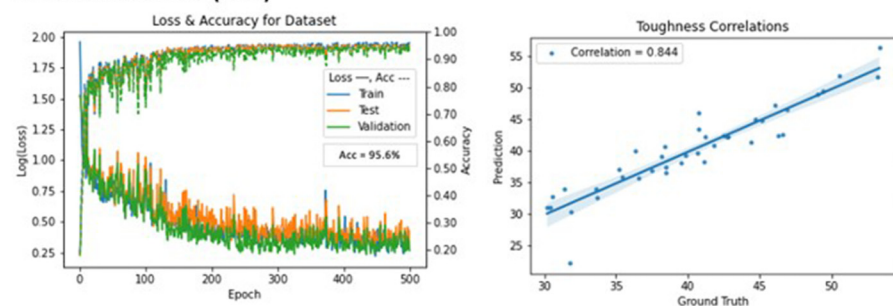
STRENGTH:**1. Full SET1 (100)****2. Full Mixed Set (200)****TOUGHNESS****3. Full SET1 (100)****4. Full Mixed Set (200)**

FIG. 5. Training and property prediction performance of the PNA model. (1) and (2) show the strength prediction for the single set of spider web segments and the mixed set of web segments, while (3) and (4) show training and predicting performance regarding spider web toughness. The plots (from left to right) are the loss in logarithmic value, and accuracy for training, testing, and validation data sets, with test accuracy labeled; the correlation between prediction value and ground truth, as well as the coefficient of determination (R^2) labeled. Loss and accuracy are calculated using MAE.

expanding the data set and tailoring hyperparameters of models accordingly.

Furthermore, PNA model degradation is assessed to substantiate its improved power in feature learning. Two degraded PNA models and their performance for strength prediction of the single web set are displayed in Fig. S4(a) in the [supplementary material](#). These models retain a three convolutional layer structure but implemented only one “mean” aggregator and one “identity” scaler, the same as configured in the GCN structure. The first model uses a “mean” aggregator and a reduced number of parameters similar to the GCN model’s, with the hidden layer dimension decreased to 15. Its best accuracy during the training process decreased from 94.4% to 92.6% since message passing is simplified, and R2 decreased from 0.87 to 0.70. The second model implements a “mean” aggregator but with 32 hidden channels kept, which similar accuracy of 92.2% and R2 = 0.76. The summary of two degraded PNA model structures and the number of parameters are shown in Fig. S4(b) in the [supplementary material](#).

C. Sensitivity analysis of graph neural networks

To optimize the model performance with the limited data set, several hyperparameters and optimization techniques are studied to compare their impacts on the existing PNA model. The prediction performance is compared regarding the training loss and testing accuracy. A total of six assessments are shown in Fig. S5 in the [supplementary material](#), including the comparison for the loss function, the implementation of BatchNorm and Dropout layer, optimizer, pooling layer, learning rate, and scheduler. First, three types of loss function are evaluated: MAE, mean square error (MSE), and Huber loss. MAE can better capture the general data feature, while MSE is more sensitive to outliers. Huber loss is a combination of MAE and MSE with a threshold value for exchanging the error function.⁵⁷ It is indicated MAE and Huber loss with a default boundary of 1 better fit the data set for model training [Fig. S5(a) in the [supplementary material](#)].

Second, the implementation of BatchNorm and the dropout layer is analyzed. The BatchNorm layer is designed for normalizing the output from mini-batches, and the Dropout layer for regularization with a defined sample probability to prevent synchronous weight adaption and increase the training efficiency.⁵⁸ As shown in Fig. S5(b) in the [supplementary material](#), the best PNA model performance is present with only the BatchNorm layer implemented, followed by none of these techniques applied, then one Dropout layer with small sample probabilities. Applying both BatchNorm and Dropout layers deteriorates the performance instead. Besides, different optimizers show perceptible impacts: Adam (stochastic optimization derived from adaptive moment estimation⁵⁴) and Rprop (adaptive optimization through backpropagation on error function⁵⁹) optimize the prediction better than Adagrad (adaptive stochastic optimization using subgradients⁶⁰) for the current model [Fig. S5(c) in the [supplementary material](#)]. While the pooling layer, initial learning rate, and scheduler for learning rate adjustments show trivial impacts on PNA model training [Figs. S5(d)–S5(f) in the [supplementary material](#)].

Except for training parameters and techniques, the implementation of edge features for data set customization is studied

as well. Both GCN and PNA models are trained with edge length incorporated as an edge feature for strength and toughness prediction for a mixed set of data (200 graphs). Compared with the original model performance without edge feature input, this GCN model indicates similar prediction performance regarding accuracy and correlation for both strength and toughness [Figs. S6(a)–S6(b) in the [supplementary material](#)]. While the PNA model shows slightly higher prediction accuracy and correlation regarding both strength and toughness [Figs. S6(c)–S6(d) in the [supplementary material](#)], yet not a significant improvement on prediction performance. A possible presumption for the little impact on edge implementation is that the edge length as an underlying feature may already be learned through the neural network since it is related to the spatial position of nodes, which has been customized as node features.

D. GraphPerceiver model training and performance

Figure 6 shows the results obtained using the GraphPerceiver transformer model; results for training (a, d), R2 over epochs (b, e), and correlation between ground truth and model prediction (c, f) for strength and toughness, respectively. The R2 values reach 0.93 and 0.78 for strength and toughness, respectively, and compare well against the graph neural network architecture, in spite of its more general formulation and flexibility for multimodal input and output data.

Compared with the GNN models, the GraphPerceiver model show more stable training processes for both strength and toughness and has the capacity to make prediction with higher correlations with current full dataset as input. Specifically, as shown in Table II, the GraphPerceiver models shows faster training loss convergence and more stable training performance, with a smoother training curve and less loss variations, for both strength and toughness prediction. Besides, the best R2 value for the strength prediction for GraphPerceiver is about 1.2 and 1.7 times higher than the GCN and the PNA model, while for toughness prediction, the PNA model performs slightly better correlation than GraphPerceiver, followed by the GCN model. It is noted that future work could explore the adaptation of further hyperparameter exploration, optimizers, or alternate learning strategy (e.g., pretraining and fine-tuning) to improve performance even further.

IV. SYNTHETIC SPIDER WEB GENERATION

In this section, a prototype of a computational optimization tool for synthetic spider web generation is discussed with the developed GNN prediction models applied for property identification. Two generation methods, nodes addition and cube composition, are developed and applied. A node addition method for web generation combined with an inference model for property prediction is processed through multi-objective optimization for Pareto front design optimization targeting web strength and ductility. While it is noted that the web generation methods used here are limited in learning the innate intelligence of spider web designs based on the trained inference models, web generation with other deep learning methods (e.g., adversarial neural nets or language models building on the GraphPerceiver) offer future research opportunities.

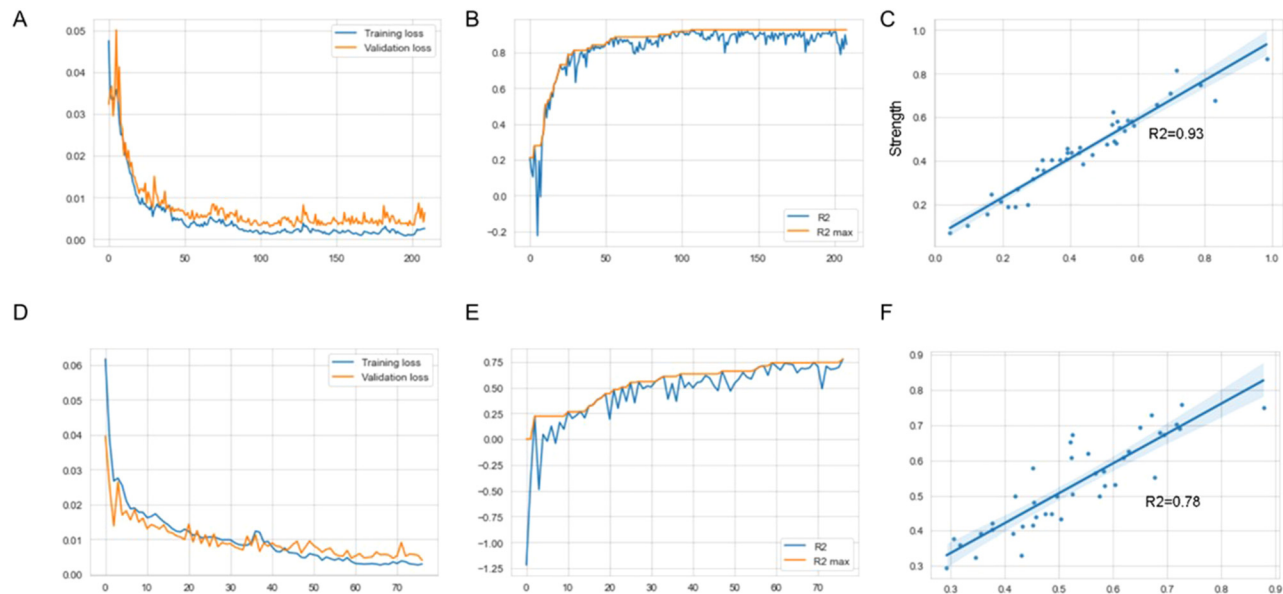


FIG. 6. Results obtained using the GraphPerceiver model; results for training (a, d), R2 over epochs (b, e), and correlation between ground truth and model prediction (c, f) for strength and toughness, respectively. The R2 values reach 0.93 and 0.78 for strength and toughness, respectively, and compare well against the graph neural network architectures studied in this paper.

A. Web generation methods

Two manual generation methods are proposed and used here: (1) Nodes Addition: randomly adding additional nodes and edge connections to original web sections and (2) Cube Composition: alternatively stacking two web cubes that are cropped from original web sections, then adding edge connections for cube connecting planes. The generated spider web geometries are stored in CSV files with the same format to be further load into the inference model for property prediction. For each generation method, four variables (Va, Vb, Vc, and Vd) are introduced for synthetic web design and used for updating the design through the next web optimization process (Table III and Fig. S7 in the supplementary material).

The specific steps of generation methods and their design variables are specifically explained below.

In the first method, (1) nodes addition, also as shown in Fig. S7(1) in the supplementary material,

1. The original web section is selected.
2. New nodes are randomly added (Va) and placed in defined space, which is the area extending from the original cube section by a certain percent of the original cube length (Vb).
3. New edges are constructed for new nodes. A certain number of nearest nodes (Vc) for each new node are searched. Then, with maximum connectivity set as four, the edge connections for new nodes will be randomly selected from one to four. New edges will be constructed between new nodes and their nearest nodes regarding their connectivity.

In the second method, (2) cube composition, also as shown in Fig. S7(2) in the supplementary material,

1. The cube sections are extracted from original web sections, and two cubes are selected.
2. The new web is constructed by alternatively stacking the two cube sections.
3. The single-connection nodes within the edge connection areas are selected. The connection area is identified by the specified percent (Va) of cube length extended from the connecting planes.
4. New edges are constructed for a certain percentage (Vb) of selected nodes. Same as the first method, new edge construction is through the nearest nodes (Vc) and their connectivity (Vd).

B. Multi-objective optimization

Combining the generation algorithm and the prediction model, a computational tool for synthetic web optimization is

TABLE II. Performance comparison between GraphPerceiver and the GNN models for strength and toughness prediction.

Performance	Property	Performance comparison (from better to worse)
Training loss	Strength	GraphPerceiver > GCN > PNA
variation	Toughness	GraphPerceiver > GCN > PNA
Prediction	Strength	GraphPerceiver > GCN > PNA
correlation (R2)	Toughness	PNA > GraphPerceiver > GCN

TABLE III. Design variables for generation methods: (1) nodes addition and (2) cube composition.

Variable	Method	
	(1) Nodes addition	(2) Cube composition
Va	The number of new nodes	% length of the cube section
Vb	% length of the original web section	% of single-connection nodes
Vc	The number of nearest nodes searched	The number of nearest nodes searched
Vd	The maximum connectivity (set as 4)	The maximum connectivity (set as 4)

tested, targeting mechanical properties as a first application. The schematic of the process is shown in Fig. 7. The nodes addition method is applied for synthetic web design, and constructed GNN models are implemented for web property identification. The MOO is performed with the Non-dominated Sorting Genetic Algorithm II (NSGA-II)⁶¹ as the algorithm for Pareto front design selection. The population size is defined as 20, and 100 generations are processed, which results in 20 designs selected among 2000 generated designs. The optimization variables are the design variables for synthetic web generation and are updated through iterations. The two optimization objectives are web strength and ductility, which are complementary mechanical properties for web

structures. In this implementation, the ductility, as the critical strain at failure, is estimated as $W/\sigma \times 2$, where W is tensile toughness and σ is the strength as the ultimate stress since the strain-stress curve of original data set forms approximately a triangular [as shown in Fig. 2(b) in Refs. 1 and 12], where the toughness represents its area under the curve, strength value as the height, and the ductility as the base of triangular. The validation of the ductility for synthetic webs can be taken as a future step.

The designs as Pareto optimal targeting strength and ductility are optimized and selected among the whole design space. Some of the optimized synthetic web designs are visualized in Fig. 8 with both final structure and addition structure indicated. The selected designs are scatter plotted [Fig. 9(1a)], and the two objectives f_1 and f_2 are negative strength and negative ductility of generated web designs correspondingly since the genetic algorithm is applied for minimizing the negative value of two targeted properties.

C. Synthetic web analysis

The edge length distribution and property relationships of both optimized web designs [Fig. 9(1)] and all generated designs [Fig. 9(2)] show similar distribution and trends as the original data sets (Fig. S1 in the supplementary material). It is also a validation of the prediction model with properly captures the trend of web properties from the original data set. The synthetic spider web design space, with all the generated webs through iterations, is

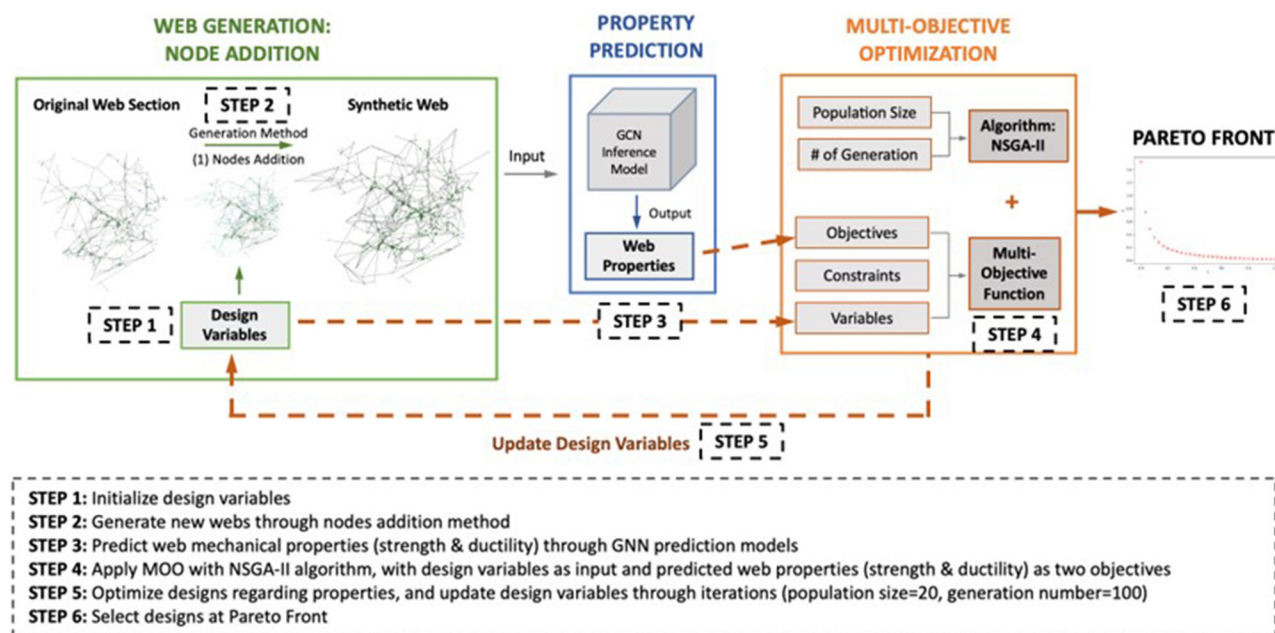


FIG. 7. Schematic of the *de novo* web generation and optimization process. The synthetic web is generated through node addition, and graph-structured web data are imported into the Inference model for property prediction. The design variable and targeted web properties (web strength and ductility) are imported as the multi-objective function for multi-objective optimization (MOO), with Non-dominated Sorting Genetic Algorithm II (NSGA-II) as the selection algorithm with population size and number of the generation defined (e.g., population size of 20 and generation number of 100 is used, to select 20 designs among 2000 designs through 100 iterations). The design variables are updated through the optimization iteration, and Pareto Front is selected as desired synthetic web structure.

(1) New Nodes: 12

(2) New Nodes: 37

(3) New Nodes: 96

(4) New Nodes: 141

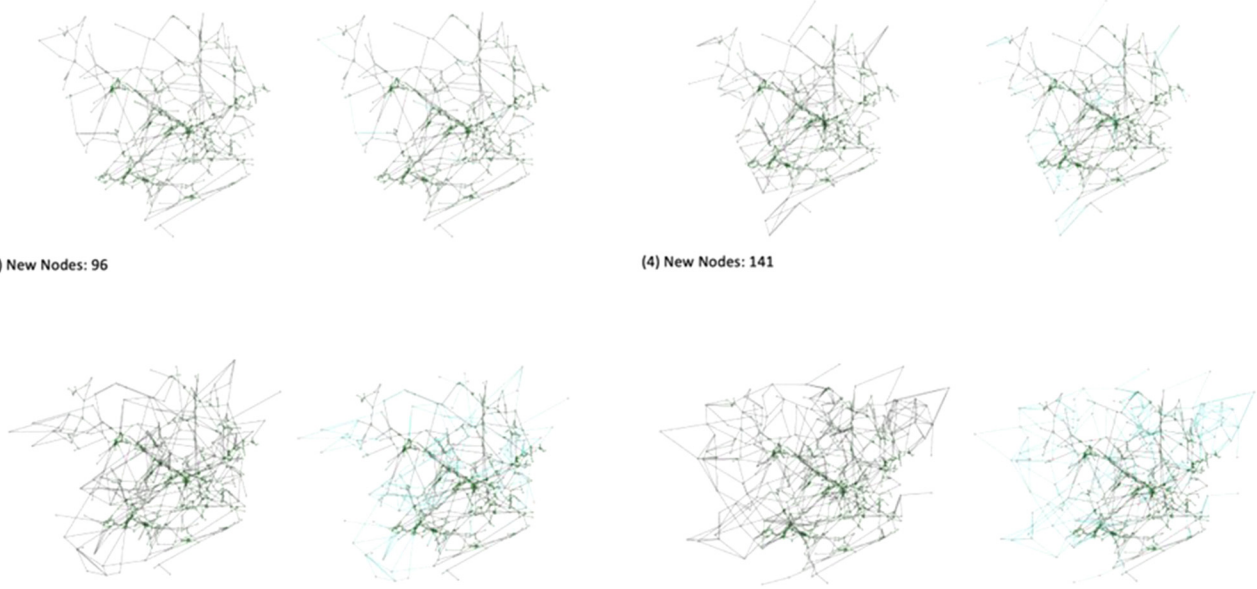


FIG. 8. Visuals of selected *de novo* webs. For each design, left one is the final geometries, and right one combines the original section (black portion) and additional structure (blue portion). [Va, Vb, Vc, Vd] with values of (1) [12, 0.38, 6, 4], (2) [37, 0.37, 6, 4], (3) [96, 0.49, 11, 4], and (4) [141, 0.5, 9, 4].

visualized in Fig. 10. Each design is noted as a point, with property values indicated through color intensity for all strength, toughness, and ductility. Three axes, x , y , and z represent three design variables (Va, Vb, and Vc, respectively, as indicated for the nodes addition method in Table III). For generated design space, ss indicated, as the number of new nodes increases, the web strength and toughness decrease, but ductility increases, which may be because the additional structure has instead inferior strength to the original web sections. When more number of new nodes are added, the lower % edge extension for node placement, the higher strength and toughness, whereas, when less number of new nodes constructed, the % edge extension has a lower impact on web property. The number of nearest nodes searched for connection has little effect on synthetic web properties. The relationship among design variables and properties is also shown in corresponding parallel coordinate plots (Fig. S8 in the supplementary material).

To better compare and analyze the synthetic webs, graph theory is applied for quantifying the resulting network structures. The graph characteristics, including degree, clustering coefficient, and density, of the original and generated webs are discussed as follows with web statistics shown in Fig. S9 in the supplementary material. The degree is presented as the average connectivity for all the nodes; the clustering coefficient is calculated as the average value of local coefficients for undirected graphs [as indicated in Eq. (3)]; and the density is defined as the ratio between the number

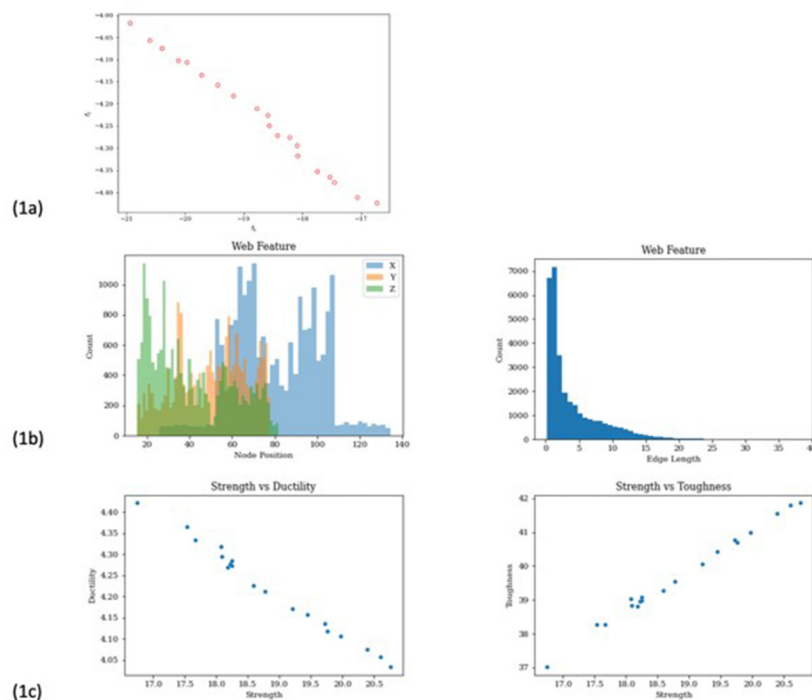
of existing edges and the maximum number of possible edges,

$$\text{Clustering Coe.} = \frac{1}{N} \sum_{i=1}^N \frac{2L_i}{K_i(K_i-1)}, \quad (3)$$

where N is the number of nodes for each graph and K_i and L_i are the degree and the number of connections among the neighboring nodes for node i . The kernel density estimate (KDE) plots of these graph characteristics for the original spider webs are shown in Fig. S9(1) in the supplementary material with the values of generated webs marked as red dots on the curve for each plot. In general, the bimodal distributions are observed with two peaks for each plot though there is less distinction for the degree distribution, since the original data set comprises two types of datasets that have relatively distinct architecture and properties (as mentioned in Sec. II B).

Furthermore, the comparisons between the original and generated spider webs for normalized relations among these graph characteristics are shown in Fig. S9(2) in the supplementary material. For the degree of new webs, some of the values are scattered between the two peaks of the original data set, and the remaining values are clustered near the maximum connectivity [Fig. S9(1) in the supplementary material]. The general design logic of spider web degree is followed, whereas higher connectivity would contribute to optimizing specific properties in this

1. Selected Webs



2. All Generated Webs

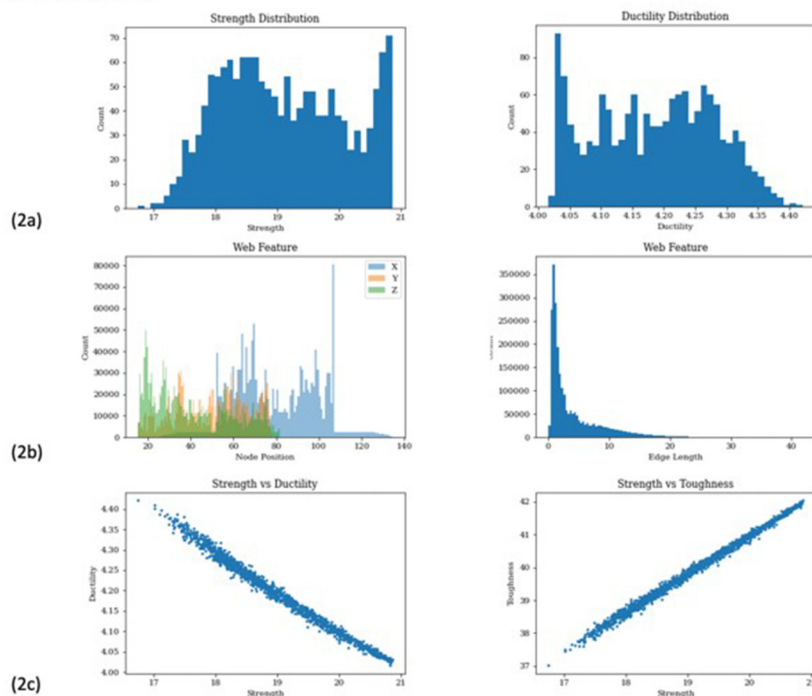


FIG. 9. Results of synthetic web generation. (1) For optimized synthetic, *de novo* spider webs: (1a) shows the Pareto front results and the selection among the design space, (1b) illustrates the distribution of web features for nodes' spatial position and edge length, (1c) the comparison of mechanical properties for synthetic webs: the strength and ductility are complementary, while the strength and toughness are in a positive relationship and generally linear. (2) For all generated spider web designs, (2a) indicates the property distribution among the design space of synthetic webs and (2b) and (2c) show the distribution of web features and property comparison, respectively.

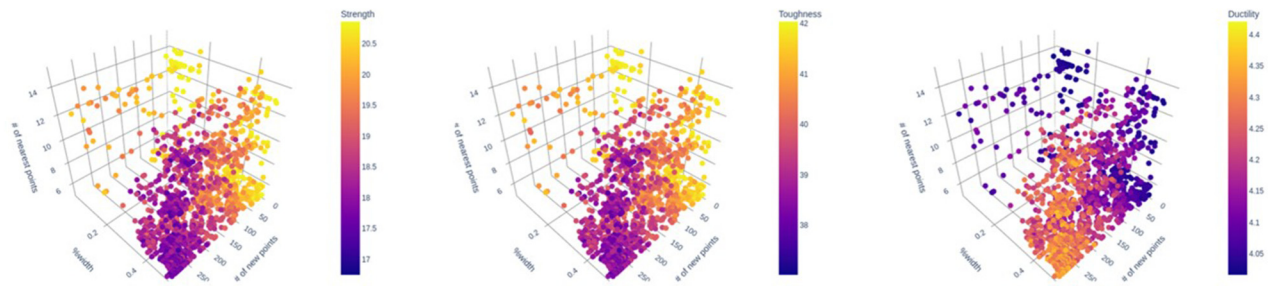


FIG. 10. Visualization of the spider web design space. Each point defines one synthetic web design, with color intensity presenting the corresponding property value for strength, toughness, and ductility from left to right. The x -, y -, and z -axes represent three design variables. As indicated for the first method, V_a , V_b , and V_c , are, respectively, the number of new nodes, the percent of web length extension for node placement, and the nearest point searched for new edge connection. In this method, as the number of new nodes (V_a) increases, the web strength and toughness decrease, but ductility increases; When more new nodes are added, the lower % edge extension for node placement, the higher strength, and toughness, whereas, when a smaller number of new nodes constructed, the % edge extension has a lower impact on web property. We find that the number of nearest nodes searched for connection has little effect on synthetic web properties.

prototype. As for the clustering coefficient of generated webs, the generated webs are distributed around the larger mode and maximum values, which indicates this optimization process leads to a relatively higher graph cohesiveness and tendency to form subsegments. The density values of new graphs are very close and are relatively higher than the mode. Higher density might be beneficial to the web properties optimized, but the current analysis could also be limited due to the small amount of data, while the augmentation would be taken as a future step. Furthermore, as shown in Fig. S9(2) in the [supplementary material](#), the generated webs follow a similar trend for the density and degree correlation as the original web graphs, which help substantiate the reliability of generated graph structure. As for the relations of graph density and clustering coefficient, though the general trend deviates, it follows the relations of the subset of the original data set (formed by blue dots on the left side of the plot).

In addition, since the ultimate strength is one of the criteria for web optimization, the weak points of the generated webs are interesting to explore, and the relative analysis is presented as follows. Uniaxial simulations are implemented to observe the nodal stress performance, and the statistic of one web is illustrated in Fig. S10 in the [supplementary material](#) as an example. The same simulation process as indicated in Sec. II A is operated, using LAMMPS for simulating uniaxial stretching in three directions (x , y , and z) and OVITO for web behavior and stress visualization with output data. Two types of nodes are identified as fixed atoms in the simulation, including the nodes with single connections and the ones near the box boundaries. Higher nodal stress indicates a higher potential for the node being broken, and Von Mises stress is collected for analysis.

As shown in Fig. S10(a) in the [supplementary material](#), normal distributions are observed for stress values among the nodes under uniaxial stretching in all x , y , and z directions. As indicated in the relational plots [Fig. S10(b) in the [supplementary material](#)] between stress and connectivity, higher stresses are mostly present in nodes with higher connectivity, especially three or four [the degree of spider webs is around three [Fig. S9(1) in the [supplementary material](#)], and lower stresses are more evenly

distributed among the nodes, since node with higher connectivity may have more loads concentrated. Furthermore, the web stretching behavior and stress value distribution under tensile loadings are visualized in Fig. S10(c) in the [supplementary material](#), and the relative weak points with higher stress values are highlighted. The weak nodes generally occur either near the center or around the boundaries under tensile loadings. Higher stress of the nodes in clustering areas may result from higher local elastic modulus applying the same strain if regarding the whole web as a material, while these nodes located in the sparser area along the stretching direction may have higher stress concentration.

V. CONCLUSIONS

A fast prediction method for mechanical properties of spider webs is reported in this paper, facilitated through graph and transformer neural networks, exploring novel ways to predict micro- and microscopic graph mechanical properties using deep learning. Training and inference models are developed using GCN and PNA architectures with graph-structured spider web data set as input and web mechanical properties as output, including web strength and toughness. A comparison with a novel GraphPerceiver model was developed and shown to achieve similar performance as the graph models. While the focus of the paper has been largely on the graph models, the excellent performance of the attention-based model offers an interesting way to more flexibly deal with a range of input and output modalities. We leave such detailed studies to future work but anticipate that further improvements to the model can be achieved by tuning hyperparameters, as well as using larger data sets, or implementing novel training strategies. Notably, even though the data available is quite small, we were successfully able to train the model and achieve good performance. Such numerical experiments may stimulate other exploration of transformer models for physical systems.

Thus far, a total of 200 digital spider web graphs and their web properties are available and were used for the training, testing, and validation of the models. The models provide overall good predictions with computational costs reduced although the current

data set is small. In general, the PNA model has higher prediction accuracy and correlation. While the GCN model indicates a more stable prediction for strength, the PNA model for toughness. The PNA model with a more complex structure and higher number of model parameters may require more data fed to improve model training from overfitting issues and further stabilize prediction performance. The current models are constructed based on a reasonable selection of model hyperparameters and optimization techniques. Besides, a prototype of a computational design for synthetic spider web generation is developed with the prediction model implemented, which also verifies the impact and the prediction capability of the constructed regression model. MOO is performed for synthetic web design, targeting web strength and ductility, with NSGA-II algorithm applied for Pareto front selection. While it is noticed these manual methods have limitations in learning the intelligence of spider web designs, the generation with deep learning method will be studied as a future step.

Using the fast prediction method proposed here, spider web structures could be studied with additional quantified properties, for web local variations and web construction process, or vibrational modes, or other material features such as optical properties. In addition, these GNN models can be implemented as a validation method for spider web studies and integrated with the transformer approaches introduced here. Future work could also consider expanding the spider web dataset for model training using the same imaging and web analysis methods¹⁰ with diverse factors considered. The PNA model parameters could be further tailored for a better prediction performance with the data set augmented. Besides, a multi-objective prediction model could be studied. Moving forward, graph generation methods through deep learning models for the spider web structure can be explored as another data augmentation and design exploration method; the latent space interpolation would be interesting for structural design and web construction study. Additionally, 3D printing and laboratory tests could be incorporated for web property checks and method validation, as well as digital simulation. These numerical modeling advances could be used as convenient and effective tools for diverse web analysis and bio-inspired structure designs. The generative algorithms explored here could further be used to create expanded data sets and add new training data to the available graph sets for future design or machine learning applications.

SUPPLEMENTARY MATERIAL

See the [supplementary material](#) for additional training and test results and analyses.

ACKNOWLEDGMENTS

We acknowledge support by ARO (No. W911NF1920098), ONR (No. N000141912375), AFOSR (No. FA9550-15-1-0514), and the IBM-MIT AI lab. Additional support from NIH (No. U01EB014976) is acknowledged.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Wei Lu: Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software (equal); Validation (equal); Visualization (equal); Writing – original draft (equal). **Zhenze Yang:** Formal analysis (equal); Investigation (equal); Methodology (equal); Writing – review and editing (equal). **Markus J. Buehler:** Conceptualization (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Resources (equal); Software (equal); Supervision (equal); Visualization (equal); Writing – original draft (equal); Writing – review and editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

- ¹D. L. Barreiro, J. Yeo, A. Tarakanova, F. J. Martin-martinez, and M. J. Buehler, “Multiscale modeling of silk and silk-based biomaterials—A review,” *19*, e1800253 (2019).
- ²M. J. Buehler and N. M. Pugno, “Hierarchical simulations for the design of supertough nanofibers inspired by spider silk,” *Phys. Rev. E* **82**, 056103 (2010).
- ³S. Olena, S. Lin, M. M. Jacobsen, D. Rizzo, D. Li, and M. Simon, “Effect of sequence features on assembly of spider silk block copolymers,” *J. Struct. Biol.* **186**, 412–419 (2014).
- ⁴I. Su, N. Narayanan, M. A. Logrono, K. Guo, A. Bisshop, and R. Mühlethaler, “In situ three-dimensional spider web construction and mechanics,” *Proc. Natl. Acad. Sci. U.S.A.* **118**, e2101296118 (2021).
- ⁵A. Y. Heckel, “Spider web geometry inspires long span roof trusses,” M.Eng. Thesis, CEE, MIT, Cambridge, May 2020 [Online]. Available: <https://dspace.mit.edu/handle/1721.1/127288>
- ⁶Z. Qin, B. G. Compton, J. A. Lewis, and M. J. Buehler, “Structural optimization of 3D-printed synthetic spider webs for high strength,” *Nat. Commun.* **6**, 7038 (2015).
- ⁷D. I. Spivak and M. J. B. Reoccurring, “Reoccurring patterns in hierarchical protein R occurring patterns in hierarchical protein materials and music: The power of analogies,” *BioNanoScience* **1**, 153–161 (2011).
- ⁸Y. Joyce *et al.*, “Materials by design: Merging proteins and music,” *Nano Today* **7**, 488–495 (2012).
- ⁹I. Su *et al.*, “Interactive exploration of a hierarchical spider web structure with sound,” *J. Multimodal User Interfaces* **16**, 71–85 (2022).
- ¹⁰I. Su *et al.*, “Imaging and analysis of a three-dimensional spider web architecture,” *J. R. Soc. Interface* **15**, 20180193 (2018).
- ¹¹T. Beleyur, T. G. Murthy, S. Singh, H. Somanathan, and D. Uma, “Web architecture, dynamics and silk investment in the social spider *Stegodyphus sarasinorum*,” *Animal Behav.* **179**, 139–146 (2021).
- ¹²E. L. Buehler, I. Su, and M. J. Buehler, “Webnet: A biomateriomic three-dimensional spider web neural net,” *Extreme Mech. Lett.* **42**, 101034 (2021).
- ¹³K. Guo, Z. Yang, C.-H. Yu, and M. J. Buehler, “Artificial intelligence and machine learning in design of mechanical materials,” *Mater. Horiz.* **8**(4), 1153–1172 (2021).
- ¹⁴G. X. Gu, C.-T. Chen, and M. J. Buehler, “De novo composite design based on machine learning algorithm,” *Extreme Mech. Lett.* **18**, 19–28 (2018).
- ¹⁵Z. Yang, C.-H. Yu, and M. J. Buehler, “Deep learning model to predict complex stress and strain fields in hierarchical composites,” *Sci. Adv.* **7**(15), eabd7416 (2021).
- ¹⁶Y.-C. Hsu, C.-H. Yu, and M. J. Buehler, “Using deep learning to predict fracture patterns in crystalline solids,” *Matter* **3**(1), 197–211 (2020).

- ¹⁷C. H. Yu, Z. Qin, F. J. Martin-Martinez, and M. J. Buehler, "A self-consistent sonification method to translate amino acid sequences into musical compositions and application in protein design using artificial intelligence," *ACS Nano* **13**, 7471 (2019).
- ¹⁸T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017—Conference Track Proceedings*, eprint arXiv:1609.02907.
- ¹⁹A. Vaswani *et al.*, "Attention is all you need," *Adv. Neural Information Process. Syst.* **2017**, 5999–6009 (2017).
- ²⁰Z. Wang, L. Zheng, Y. Li, and S. Wang, "Linkage based face clustering via graph convolution network," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, 2019), pp. 1117–1125.
- ²¹L. Yang, D. Chen, X. Zhan, R. Zhao, C. C. Loy, and D. Lin, "Learning to cluster faces via confidence and connectivity estimation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, 2020), pp. 13369–13378.
- ²²L. Yang, X. Zhan, D. Chen, J. Yan, C. C. Loy, and D. Lin, "Learning to cluster faces on an affinity graph," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, 2019), pp. 2293–2301.
- ²³S. Vashishth, M. Bhandari, P. Yadav, P. Rai, C. Bhattacharyya, and P. Talukdar, "Incorporating syntactic and semantic information in word embeddings using graph convolutional networks," in *ACL 2019—57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, eprint arXiv:1809.04283.
- ²⁴L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, no. 01 (2019), pp. 7370–7377.
- ²⁵A. Cetoli, S. Bragaglia, A. D. O'Harney, and M. Sloan, "Graph convolutional networks for named entity recognition," preprint arXiv:1709.10053 (2017).
- ²⁶D. Marcheggiani, J. Bastings, and I. Titov, "Exploiting semantics in neural machine translation with graph convolutional networks," in *NAACL HLT 2018—2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies—Proceedings of the Conference*, eprint arXiv:1804.08313.
- ²⁷L. Wu, J. Li, P. Sun, R. Hong, Y. Ge, and M. Wang, "Diffnet++: A neural influence and interest diffusion network for social recommendation," in *IEEE Transactions on Knowledge and Data Engineering* (IEEE, 2020), pp. 1–14.
- ²⁸L. Wu, R. Hong, P. Sun, X. Wang, Y. Fu, and M. Wang, "A neural influence diffusion model for social recommendation," in *SIGIR 2019—Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, New York, NY, USA, pp. 235–244.
- ²⁹J. Tao *et al.*, "MVAN: Multi-view attention networks for real money trading detection in online games," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, NY, USA, 2019), pp. 2536–2546.
- ³⁰J. D. Acevedo-Viloria *et al.*, "Relational graph neural networks for fraud detection in a super-app environment," in *KDD-MLF 2021, August 14–18, 2021, Virtual Workshop*, eprint arXiv:2107.13673.
- ³¹J. X. Zhong, N. Li, W. Kong, S. Liu, T. H. Li, and G. Li, "Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, 2019), pp. 1237–1246.
- ³²S. Wang *et al.*, "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning," preprint arXiv:2111.04333 (2021).
- ³³H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler, "Fast interactive object annotation with curve-GCN," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE, 2019), pp. 5252–5261.
- ³⁴Q. Zhang, Q. Jin, J. Chang, S. Xiang, and C. Pan, "Kernel-weighted graph convolutional network: A deep learning approach for traffic forecasting," in *Proceedings—International Conference on Pattern Recognition* (IEEE, 2018), 1018–1023.
- ³⁵M. Lv, Z. Hong, L. Chen, T. Chen, T. Zhu, and S. Ji, "Temporal multi-graph convolutional network for traffic flow prediction," *IEEE Trans. Intell. Trans. Syst.* **22**(6), 3337–3348 (2021).
- ³⁶K. Osanlou, C. Guettier, A. Bursuc, T. Cazenave, and E. Jacopin, "Constrained shortest path search with graph convolutional neural networks," preprint arXiv:2108.00978 (2021).
- ³⁷M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics* **34**(13), i457–i466 (2018).
- ³⁸T. Nguyen, H. Le, T. Le, P. Quinn, T. Nguyen, T. D. Le, S. Venkatesh, "GraphDTA: predicting drug–target binding affinity with graph neural networks," *Bioinformatics* **37**(8), 1140–1147 (2021).
- ³⁹V. Gligorijević *et al.*, "Structure-based protein function prediction using graph convolutional networks," *Nat. Commun.* **12**(1), 3168 (2021).
- ⁴⁰S. Ryu, J. Lim, S. H. Hong, and W. Y. Kim, "Deeply learning molecular structure–property relationships using attention- and gate-augmented graph convolutional network," preprint arXiv:1805.10988 (2018).
- ⁴¹D. M. Gysi *et al.*, "Network medicine framework for identifying drug-repurposing opportunities for COVID-19," *Proc. Natl. Acad. Sci. U.S.A.* **118**(19), 1–11 (2021).
- ⁴²X. Xiang, Y. Chen, J. Gao, P. Zhong, H. Song, and J. Gao, "Cropping graph convolution neural network for prediction of compound carcinogenicity," in *ICCCSE 2021—IEEE 16th International Conference on Computer Science and Education* (IEEE, 2021), pp. 864–869.
- ⁴³Y. Ding, X. Jiang, and Y. Kim, "Relational graph convolutional networks for predicting blood-brain barrier penetration of drug molecules," *Bioinformatics* **36**(2), 211 (2022).
- ⁴⁴H. Stärk *et al.*, "3d infomax improves gnns for molecular property prediction," in *International Conference on Machine Learning* (PMLR, 2022), Vol. 162, pp. 20479–20502.
- ⁴⁵K. Guo and M. J. Buehler, "A semi-supervised approach to architected materials design using graph neural networks," *Extreme Mech. Lett.* **41**, 101029 (2020).
- ⁴⁶E. Whalen and C. Mueller, "Toward reusable surrogate models: Graph-based transfer learning on trusses," *J. Mech. Design* **144**(2), 021704 (2022).
- ⁴⁷G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Velickovic, "Principal neighbourhood aggregation for graph nets," *Adv. Neural Information Process. Syst.* **2020** (2020).
- ⁴⁸J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *35th International Conference on Machine Learning, ICML 2018* (PMLR, 2018), Vol. 80, pp. 5708–5717.
- ⁴⁹I. Su and M. J. Buehler, "Mesomechanics of a three-dimensional spider web," *J. Mech. Phys. Solids* **144**, 104096 (2020).
- ⁵⁰A. P. Thompson *et al.*, "LAMMPS—A flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales," *Comput. Phys. Commun.* **271**, 108171 (2022).
- ⁵¹S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *J. Comput. Phys.* **117**, 1–19 (1995).
- ⁵²M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *ICLR* **1**, 1–9 (2019).
- ⁵³A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 32, pp. 8024–8035 (2019).
- ⁵⁴D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings*, eprint arXiv:1412.6980.
- ⁵⁵A. Jaegle *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," preprint arXiv:2107.14795 (2021).
- ⁵⁶P. Virtanen *et al.*, "SciPy 1.0—Fundamental algorithms for scientific computing in Python," *Nat. Methods* **17**, 261–272.
- ⁵⁷P. J. Huber, "Robust estimation of a location parameter," *Ann. Math. Stat.* **35**(1), 73–101 (1964).

⁵⁸G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," preprint arXiv:1207.0580 (2012).

⁵⁹M. Riedmiller and H. Braun, "Direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *1993 IEEE International Conference on Neural Networks* (IEEE, 1993), pp. 586–591.

⁶⁰J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Proc. IEEE Conf. Decision Control* **12**, 5442–5444 (2012).

⁶¹K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evolutionary Comput.* **6**(2), 182–197 (2002).