ABSTRACT Actin cytoskeleton morphology is able to affect and reflect the cellular mechanical properties. However, due to the lack of efficient approaches to quantifying actin cytoskeleton and mechanical properties, the study of mechanotransduction dynamics is still laborious. In this paper, a model to characterize the cellular actin cytoskeleton morphology was built using the graph to vector embedding technique together with neural network in machine learning (ML). The proposed ML model consists of a skip-gram model followed by a fully connected classifier. The images of NIH/3T3 cells treated with Latrunculin B at different concentrations were taken as the inputs, and the outputs were the actin cytoskeleton morphology labels defined by treatment concentrations (i.e., the actin depolymerization level). The proposed model was also compared to a general convolutional neural network (CNN) and three commonly used transfer learning models (GoogleNet, Xception, and VGG16), the results demonstrated the capabilities of the proposed model in extracting actin cytoskeleton features, avoiding overfitting, and keeping the model generalization.

INDEX TERMS Actin cytoskeleton, classification, graph to vector embedding, machine learning, skip-gram model.

I. INTRODUCTION
A number of studies have shown that the cellular cytoskeleton can significantly affect and reflect the numerous biomechanical properties of cells [1], [2]. Particularly, the cytoskeleton reshapes its morphology to regulate the cellular dynamic imbalance triggered by outside stimuli, such as external force, substrate stiffness change, and drug treatment [3], [4], [5]. The actin cytoskeleton, as one of the main kinds of cytoskeletal filaments, is able to control the cellular mechanotransduction properties, such as elasticity, poroelasticity and focal adhesion through morphology reorganization to affect cellular motility, contractility, and dynamic stability [1], [2]. Therefore, the actin cytoskeleton morphology represents essential information of the cellular biomechanical properties. However, effectively recognizing and quantifying the morphological shape of actin cytoskeleton remains challenging, let alone investigating the correlation between the actin cytoskeleton morphology and cellular biomechanical properties.

In earlier research, Karlon et al. proposed an automated method using intensity gradient in images of endothelial cells to quantify the cells’ orientation distribution and cytoskeletal filament organization [6]. Even this method provides rapid and accurate quantification results compared to pure manual measurements, it still needs great human effort. More effort was then devoted to the related studies. For plant cells, Higaki et al. developed an image analysis framework to quantify the cytoskeleton orientation, bundling, and density using the measurement of fluorescence microscopic images [7]. Kimori et al. applied a mathematic method to quantify the morphology of biological structures of the actin cytoskeleton in the plant cells [8]. For animal cells, Alizadeh et al. proposed a way to distinguish the high metastatic and low metastatic osteosarcoma cancer cell with high accuracy using Zernike moments and geometric parameters as a measure.

The associate editor coordinating the review of this manuscript and approving it for publication was Filbert Juwono.
of cell shape [9], and they improved the proposed method with a toolbox, TISMorph, that calculates a set of quantitative measures to address the cytoskeleton state [10]. We also proposed an image recognition-based approach to quantify the actin cytoskeleton using Canny/Sobel edge detector, Hough transform, and Matlab filling tools [11]. However, these methods require manual processing and analysis on cellular cytoskeleton images, which are inefficient in dealing with a large number of cell samples and hard to deploy the quantification operation using traditional parameters. Recent advances in image modeling with neural networks have provided a way to derive representations of cell shapes and cellular structures.

Osokin et al. propose a novel application of Generative Adversarial Networks (GAN) to the synthesis of cell images taken by fluorescence microscopy [12]. HM Rostam et al. set up five supervised learning methods, i.e., support vector machine (SVM), k-nearest neighbor (kNN), Naive Bayes, logistic regression, and a random forest classifier, to build classifiers for image-based segmentation of immune cell data [13]. Lam et al. applied the support vector machine learning algorithm to distinguish between elongated and circular data [14]. Lam et al. et al. introduced the graph neural network (CNN) using the self-supervised method to learn feature representations of single cells in fluorescence microscopy images without labeled training data [15]. Jude Phillip et al. built an unsupervised machine learning method to analyze the cell and nuclear morphology of fluorescent images using the Visually Aided Morphotyping Image Recognition (VAMPIRE) algorithm, which was designed by Wu et al. [16], it offers a direct and quantitative way of measuring the extent of morphological heterogeneity within cell populations efficiently [17]. However, these methods either focus on external cell morphology and cell cycling phase [12], [13], [14], [15] or require a high-throughput cell imaging platform (htCIP) that provides access to extracting high-content cellular and nuclear morphology of individual cells [17], thus cannot be applied to quantify the cellular actin cytoskeleton using the normal computation setups. Therefore, this study aims to build a new efficient deep learning model for cellular actin cytoskeleton identification through feature representations extraction, which can be further used to build the correlation between actin cytoskeleton morphology and cellular biomechanical properties using experimentally measured cellular mechanics data.

In our research, a model to characterize the cellular actin cytoskeleton morphology was built using the graph to vector embedding technique together with the neural network in machine learning. The proposed ML model consists of a skip-gram model followed by a fully connected classifier. The images of NIH/3T3 cells treated by Latrunculin B at different concentrations were taken as the inputs, and the outputs were the actin cytoskeleton morphology labels defined by treatment concentrations (i.e., the actin depolymerization level). In the ML model, the embedding tool outputs the embedded vectors of the cytoskeleton graphs, and then the embedded vectors are used by the fully connected layer to perform cytoskeleton classification. The test loss of 0.6313 and accuracy of 80.49% were obtained at the end showing that both structure and quantity features were extracted efficiently from the actin cytoskeleton images. Compared to the result presented in the previous work [18], more sample images were used to reduce the effect of overfitting and to improve the model reliability. To validate the model, the proposed model was compared to a general convolutional neural network (CNN) and three commonly used transfer learning models (GoogleNet, Xception, and VGG16). In a number of studies, the CNN and transfer learning models showed their ability to develop an internal representation of a two-dimensional image. The comparison results demonstrated the advanced capabilities of the proposed model in extracting actin cytoskeleton features, avoiding overfitting, and keeping the model generalization.

II. DEEP LEARNING FRAMEWORK FOR ACTIN CYTOSKELETON MODELING

A. CONVERT IMAGES TO GRAPHS

Actin cytoskeleton of single cells was cropped from the fluorescent cell images. The cells that overlapped with others were discarded during the data preparation. To quantify the brightness of the actin cytoskeleton, the cropped RGB images were converted to grayscale with the range of brightness for each pixel from 0 ~ 255 [19]. To remove the effect of the background color, the brightness of pixels outside the cropped area was mandatorily set as 0. To minimize the noise effect, the pixels with brightness lower than the average brightness were set to 0 [20]. Since the graph of the actin cytoskeleton needs to be extracted, a Canny edge detector was applied to skeletonize the actin fibers in images. For the skeletonized images, each non-zero pixel (i.e., the pixel with brightness larger than the average brightness) was treated as a node with its brightness as the node feature. Graphs were explored from one randomly selected root node through four directions (i.e., up, down, left, and right) using Breadth-first search and edges created between two adjacent nodes. The graph \( G \) consists of the nodes and the edges that represent the architecture and intensity features of actin cytoskeleton (see Fig. 1).

B. SAMPLING AND RELABELING ALL SUBGRAPHS FROM THE GRAPH

A subgraph is a set of nodes that appears around the selected root node \( n \) within a designed distance (or walk depth) from the root node, see Fig. 2. Weisfeiler-Lehman (WL) relabeling process was applied to sample and label the subgraphs [21], which lays the basis for the WL kernel [21], [22]. The subgraph extraction process is shown in Algorithm 1. The algorithm takes a given graph \( G \) from which the subgraph needs to be extracted, a randomly selected root node \( n \), and
FIGURE 1. Preprocessing of the fluorescent actin cytoskeleton image. A cell is cropped from the original image. Graph nodes are generated and labeled based on the pixel brightness. The list, consists of the edges between neighbor nodes in four directions and the features which are the node brightness, represents the actin cytoskeleton graph.

FIGURE 2. Subgraphs $s_{g_n}$ of node $n = 7$ with walk depth $d$ in range from $0 \sim 2$.

class GetSubgraph:
    def __init__(self, n, G, d):
        self.n = n
        self.G = G
        self.d = d

    def GetSubgraph(n, G, d):
        if walk depth $d = 0$:
            $S_{g_n}^d = 7$
        if walk depth $d = 1$:
            $S_{g_n}^d = 7$ 
               4-6-8-12
        if walk depth $d = 2$:
            $S_{g_n}^d = 7$ 
               4-2-5-6-11-8-5-13-12-11-13

        if walk depth $d = 3$:
            $S_{g_n}^d = 7$ 
               4-2-5-6-11-8-5-13-12-11-13

        return $s_{g_n}^d$

Algorithm 1 GetSubgraph

Input:
- $n$: The root node of the subgraph
- $G = (N, E, \lambda)$: Given graph
- $d$: Walk depth considered in subgraph extraction

Output:
- $s_{g_n}^d$: Rooted subgraph of walk depth $d$ around node $n$

C. GRAPH EMBEDDING

To train the deep learning model, graphs need to be converted to numerical data. One-hot encoding is one of the most popular techniques of encoding categorical data. However, one-hot encoding suffers the curse of dimensionality for high cardinality since its feature can really blow up quickly [23]. In these cases, to reduce the dimension of input data, graphs of actin cytoskeleton were embedded in vectors based on the cytoskeletal morphology (i.e., graph structure) and quantity (i.e., fluorescent brightness). The idea of the graph embedding is to train the weight matrix of the hidden layer to find efficient representation for given graphs [24] (see Fig. 3). Note that the input of the embedding could be replaced by other encoding approaches, such as a one-hot hashing trick [25], without significantly affecting the embedding model structure. In addition, deep learning APIs, such as Keras, also provide embedding models with non-fixed input lengths.

In this work, to embed the actin cytoskeleton graphs, the skip-gram (SG) model was applied as the embedding layer, which is commonly used in Word2Vec embedding and yields the highest overall accuracy, and consistently produces the highest accuracy on semantic relationships [26], [27], [28].
The skip-gram model was trained to maximize the probability of predicting subgraphs that exist in the graph that needs to be embedded on the input (see Algorithm 2). In detail, the SG model is to predict the subgraphs for each given graph $G_i$ where $G_i \in G$, and minimize

$$ J(\Phi) = - \frac{1}{|G|} \sum_{i=1}^{|G|} \sum_{n \in N_i} \log Pr(sg^{(d)}_{n,i} | \Phi(G_i)), $$

on a detected subgraph set $SG_{vocab} = \{sg_1, sg_2, \ldots\}$, where $sg^{(d)}_{n,i}$ denotes the achieved subgraphs at the node $n$ in a given graph $G_i$ with depth of $d$. The probability to predict subgraphs is estimated by

$$ Pr(sg^{(d)}_{n,i} | \Phi(G_i)) = \frac{\exp(V_{sg^{(d)}_{n,i}}^T V_{G_i})}{\sum_{sg^{(d)}_{n,j} \in SG_{vocab}} \exp(V_{sg^{(d)}_{n,j}}^T V_{G_i})}, $$

(2)

where $V_{G_i} = \Phi(G_i)$ is the embedding for $G_i$, and $V$ and $V'$ refer to input and output embeddings, respectively. The training processing of the SG model is thus to minimize $J(\Phi)$ over a graph set iteratively. In the beginning, the weight matrix $\Phi$ (i.e., embedding vector) was initialized with a random value. The given graph $G_i \in G$ was fed as a one-hot vector [29] in $e$ epochs to fit the subgraphs contained in graph $G_i$. During the embedding process, actin cytoskeleton images with similar structure and quantity were embedded closer than those without the similarities. Specifically, cell images taken under the same conditions were represented using vectors having closer distance in vector space, which improves the later classification accuracy.

Since the subgraphs are extracted from each node of each graph with the initial depth $d$, the size of $SG_{vocab}$ is very large. This means for each input, weight updates only make very small changes to a huge amount of weights even though there exists just one true example. This makes the training process very inefficient. Thus we used negative sampling which approximates the loss from the softmax layer by updating a small subset of all the weights at once – update the weights of the correct label but only a small number of the incorrect label [30]. This makes the network training more efficient.

### D. Classification

After we obtained the embedding vector of the actin cytoskeleton graph, a neural network (NN) classifier was built to identify the cytoskeleton class. Linear neurons together with rectified linear unit (RELU) activation function were taken as the learning units. Softmax was selected as the last
activation function to get the probability distribution. The dropout technique was applied to prevent overfitting [31]. Cross entropy was taken as the loss function to update the weights and hyper-parameters in the network [32]. The training loss, which needs to be minimized, is quantified by cross-entropy as
\[ H(p, q) = -\sum_{x \in X} p(x) \log q(x), \]
where \( q(x) \) is the predicted probability distribution, \( p(x) \) is the true probability distribution (i.e., true label).

To find the optimal hyper-parameters of the model which results in the best predictions, the grid search exhaustively generates candidates from a grid of specified parameter values [33]. The details of the architecture and hyper-parameter selection of the classifier are described in the following section.

III. EXPERIMENTAL DETAILS
A. CELL PREPARATION
1) CELL CULTURE
Primary mouse embryonic fibroblast cells (i.e., NIH/3T3 cell) were taken as the sample cells in this study. They were maintained at 37°C in an incubator with the humidified atmosphere of 5% CO\(_2\) in Dulbecco’s Modified Eagle’s Medium (ATCC, Rockville, MD, USA), and cultured in the medium mixed with 10%(V/V) calf bovine serum (Sigma-Aldrich, St. Louis, MO, USA) and 1%(V/V) penicillin-streptomycin (Gibco, Grand Island, NY, USA). Sample cells were seeded in 35 mm tissue culture dishes (Azzota Scientific, DE, USA), and then treated after 24 hours.

2) ACTIN CYTOSKELETON TREATMENT
To observe the actin cytoskeleton with different morphology states, the cells were treated by Latrunculin B (Millipore Sigma, Billerica, MA, USA), a commonly known actin depolymerizer, at the concentration of 0 nM, 40 nM, and 100 nM, respectively, in the aforementioned cell culture medium and incubated for 30 mins. The stock solution was made by dissolving latrunculin B in Dimethyl sulfoxide (Sigma Aldrich, St. Louis, MO, USA).

3) ACTIN CYTOSKELETON STAINING
To investigate the actin cytoskeleton, cells were fixed for 10 mins using 4% paraformaldehyde (Alfa Aesar, Ward Hill, MA, USA) in PBS, and permeabilized for another 10 mins using 0.1% Triton-X (Fisher Scientific, Fair Lawn, NJ, USA) at room temperature. The actin cytoskeleton was then stained using Actin-stain\(^7\) M 488 phalloidin (Cytoskeleton Inc, Denver, CO, USA) at the concentration of 100 nM in PBS and kept in the dark at room temperature for 30 mins.

4) FLUORESCENCE MICROSCOPE
The fluorescent actin cytoskeleton was imaged using an inverted optical microscope (IX73, Olympus, Japan). The microscope is integrated with a sola light engine (Lumencor, Beaverton, OR, USA) that offers access to solid-state illumination. To prevent the fluorescence bleaching effect and obtain the images under identical measurement conditions, all images were taken in 5 s with the same light strength and exposure time.

B. MODEL ARCHITECTURE AND SETTING
1) DATA PREPARATION
As aforementioned, NIH/3T3 cells were treated with Latrunculin B with different treatment concentrations (TC), 2036, 1555, and 1674 fluorescent images of the actin cytoskeleton were taken under Latrunculin B treatment at concentrations of 0 nM, 40 nM, and 100 nM, respectively. These three selected concentrations correspond to 0%, 62%, and 100% of actin cytoskeleton depolymerization, respectively, according to the previous study [34]. Therefore, the actin cytoskeleton classes were defined in three classes. Class 0 represents the untreated status (depolymerization = 0%), Class 1 represents the mid-treated status (depolymerization = 62% relative to the untreated case), and Class 2 represents completely treated status (depolymerization = 100% relative to the untreated case). For each image, at least 1 cell was cropped from it. In the end, total of 7371 images were preprocessed as aforementioned. 2850, 2197, and 2324 cells for class 0, class 1, and class 3, respectively, were converted to actin cytoskeleton graphs.

| Class Label | Data Size | TC  | Depolymerization |
|-------------|-----------|-----|-----------------|
| 1           | 2830      | 0 nM | 0%              |
| 2           | 2197      | 40 nM | 40%             |
| 3           | 2324      | 100 nM | 100%            |

2) EMBEDDING MODEL
For the embedding model, the target is to embed the actin cytoskeleton graph into a vector with a length of 128. Therefore, the hidden layer of the embedding model was represented by a weight matrix with 7371 rows (one for every graph in our dataset) and 128 columns (one for every hidden neuron). So the end goal of all of this is just to learn this hidden layer weight matrix. The 1 x 128 graph vector then gets fed to the output layer to fit its corresponding subgraphs. The output layer is a softmax regression classifier. Specifically, each output neuron has a weight vector that multiplies against the graph vector from the hidden layer, then the output neuron applies the softmax function to the results.

3) CLASSIFICATION MODEL
For the classifier, the input layer had 128 neurons (equals the vector dimension). Two fully connected hidden layers with neuron numbers of 32 and 128, respectively, were developed, ReLU was chosen as the activation for each neuron. One Dropout layer was applied between these two hidden layers with a dropout rate of 0.5. Since the graphs needed to be
classified into three classes, the output layer was built using 3 neurons together with the softmax function as its logistic function. The output was the probability distribution of the 3 classes. To train, validate (tune the hyperparameters of the trained model), and test (evaluate the accuracy) the proposed framework, the total images were randomly split into three portions (i.e., the training, the validation, and the test datasets) at fraction of 0.8, 0.1, and 0.1, respectively, with no overlap among the three sets. Therefore, the size of the training dataset, validation dataset, and test dataset were 4212, 527, and 526, respectively. To get the optimal hyper-parameters, grid search was applied in the training process. We specified one grid to be explored: the one with optimizers of Adam and RMSProp, initializer of Glorot uniform, epoch number in [40, 50, 60, 70], batch size in [15, 20, 25, 32], and learning rate in [0.01, 0.02, 0.05, 0.1] [35]. All parameter combinations were exhaustively explored. The training loss, training accuracy, and validation accuracy were recorded to help to tune the hyper-parameters in the classification model.

IV. RESULT AND DISCUSSION
A. GRAPH TO VECTOR EMBEDDING

Cell images were preprocessed to extract the actin cytoskeleton graphs. The obtained graphs were embedded to vectors through the skip-gram model. Graph $G_i \in \mathcal{G}$ was fed as one-hot vectors to the embedding model. In this phase, the skip-gram model was trained to maximize the probability of predicting subgraphs that exist in the graph $G_i$. The number of the input neurons $R = 7371$ was equal to the size of the graph set $\mathcal{G}$, the number of embedding neurons in the embedding layer was equal to the designed embedding vector dimension $C = 128$, and the number of output neurons was equal to the size of total subgraph set $SG_{\text{vocab}}$. The shape of the weight matrix was $(R, C)$. The embedded vectors are shown in Table 2.

As aforementioned, each graph was embedded into a vector space with the dimension of 128. To visualize the embedding result, 3-dimensional principal component analysis (3D-PCA) was applied for increasing the interpretability and minimizing the information loss at the same time [36]. 3D-PCA projects the embedded vectors onto a 3-dimensional hyperplane selected by preserving the maximum variance, thus it will most likely lose the least information. The embedded vectors are shown in Fig. 4, in which the 3D distribution of embedded vectors indicates that the vectors that representing the graphs extracted from actin cytoskeleton indeed have their own characteristic features under different treatment concentrations. The explained variance ratio of reducing vector dimension from 128 to 3 is 97.96%, which shows 3D-PCA caught most of the variance in the original data, so it is reasonable to assume that a strong strength of association lies between the original data and the 3 principal components of the 3D hyperplane. Therefore, the embedding layer efficiently reduced the graphs without missing their structure and node features. A more significant distinction could be generated in higher dimension space.

B. CLASSIFIER TRAINING AND VALIDATION

Grid search was applied to find the optimal hyperparameters. The estimator with Adam optimizer, Glorot uniform initializer, epoch number of 70, batch size 20, and learning rate at 0.05 gave us the best fitting result. Then the embedded vectors were fed to the classifier to fit their corresponding labels. The loss in the training process is shown in Fig. 5 (A). As it can be seen, the training loss is stably decreasing from 1.8939 to 0.5922, and the validation loss is decreasing from 1.416 to 0.5033. The accuracy in the training process is shown in Fig. 5 (B), the training accuracy is increasing from 46.16% and 80.29% and the validation accuracy is increasing from 62.64% to 85.44%. Thus, the performance of the validation is better than that of the training process. This happens because we used Dropout to perform the learning regularization, of which the behavior of training and validation were different. During the training process, a percentage (50%) of the features were set to zero in the case since the dropout rate was 0.5. However, during the validation, all features were

![FIGURE 4. Projecting embedded vectors to 3D space using PCA in four directions.](image-url)

| No. | Class Label | $x_0$ | $x_1$ | ... | $x_{127}$ |
|-----|-------------|-------|-------|------|-----------|
| 1   | 1           | -0.6151 | 0.9173 | ... | -0.41723  | -0.1682  |
| 2   | 1           | 1.4089  | -0.4357 | 1.8118 | 0.8886    | -0.3929  |
| 3   | 2           | 0.2851  | 0.0359 | 0.4484 | 0.1463    |          |
| ... | ...         | ...    | ...   | ...   | ...       |          |
| 7371| 3           | 0.1143  | 0.2257 | 0.4484 | 0.1463    |          |

TABLE 2. Embedded vectors.
used and scaled appropriately. So the model in the validation process was more robust, therefore, it can lead to higher accuracy. Moreover, the training loss is the average of the losses over each batch of the training data. Because the model was changing over time, the loss over the first batches of an epoch is generally higher than over the last batches. On the other hand, the validation loss for an epoch is computed using the model as it is at the end of the epoch, which results in a lower loss.

C. MODEL TEST

The actin cytoskeleton image data was split into training, validation, and test sets, and there was no overlap between the training and test sets. The reason of such arrangement is that the data independence could minimize the effects of data discrepancies and lead to better understandings of the characteristics of the model. In other words, the final test accuracy evaluation is reliable. As shown in Fig. 5 (B), the final test loss is 0.6313, and test accuracy is 80.49%. Note that the test accuracy is slightly decreased compared to our preliminary work (85.5%) [18]. This is because that a much large data size (5262 vs. 7371) is used in this work, which directly induced larger data variations. The accuracy could be further improved by removing the data noise and including more structure details in the learning process.

To further analyze the model performance, the confusion matrix is shown in Fig. 6, in which each entry denotes the number of predictions made by the model for actin cytoskeleton classification. Specifically, the entries with the same row and column indices are correct predictions of each class, and those with difference row and column indices denotes incorrect predictions. Moreover, we calculated the precision, recall, and F1-score for all classes (see Table 3).

TABLE 3. Prediction measurements.

| Class Label | Precision | Recall | F1-score |
|-------------|-----------|--------|----------|
| 1           | 0.800     | 0.743  | 0.771    |
| 2           | 0.856     | 0.796  | 0.825    |
| 3           | 0.742     | 0.905  | 0.816    |

Precision is the fraction of true positive among all the predicted positive, it reflects how many selected items are relevant [37]. Recall is the ratio of correctly predicted positive observations to all observations in the actual class, which reflects how many relevant items are selected [37]. F1-Score is the weighted average (β = 0.5) of Precision and Recall [37]. These measures also reinforce the generalization ability of the proposed model.

For each class (i.e., actin cytoskeleton depolymerization degree), the corresponding cell mechanical properties, such as Young’s modulus, could be pre-measured using Atomic Force Microscope, and then saved in the database. For example, the Young’s modulus of NIH/3T3 cells obtain subject to similar actin cytoskeleton depolymerization as in this work have been reported in previous study [34]. Therefore, the cellular elasticity of NIH/3T3 cells can be directly obtained/predicted based on the actin cytoskeleton classification result generated by the proposed model. This implies that the labor intensive cellular mechanical quantification experiments can be saved by using the proposed deep learning classification model, which could significantly improve research efficiency and productivity.
FIGURE 7. Model (A1∼A4) loss and (B1∼B4) accuracy of general CNN, GoogleNet, Xception, and VGG16, respectively, during the training and test processes. For CNN (A1 & B1), the model loss decreases from 1.223 and 1.225 to 0.825 and 0.901 for training and validation, respectively. Its accuracy increases from 46.7% and 58.1% to 72.3% and 71.6% for training and validation, respectively. The test loss is 0.912, and the test accuracy is 0.735%. For GoogleNet (A2 & B2), the model loss decreases from 1.300 and 1.092 to 0.843 and 0.953 for training and validation, respectively. The model accuracy increases from 53.4% and 59.8% to 67.8% and 66.1% for training and validation, respectively. The test loss is 1.233, and the test accuracy is 59.5%. For Xception (A3 & B3), the model loss decreases from 0.949 and 0.841 to 0.655 and 0.750 for training and validation, respectively. Its accuracy increases from 58.1% and 63.6% to 74.0% and 69.3% for training and validation, respectively. The test loss is 0.900, and the test accuracy is 66.8%. For VGG16 (A4 & B4), its loss decreases from 0.945 and 0.680 to 0.624 and 0.632 for training and validation, respectively. Its accuracy increases from 62.1% and 73.4% to 76.1% and 74.9% for training and validation, respectively. The test loss is 0.710, and the test accuracy is 73.9%.
D. CLASSIFICATION RESULT COMPARISON
To further validate the proposed graph embedding+classifier model, we compared its prediction results with that of the convolutional neural network (CNN) and transfer learning models, respectively, together with a classifier. The cropped single-cell images were taken as the inputs, and the corresponding class labels were taken as the outputs. Data of the three classes were split into training, validation, and test sets at the fraction of 0.8, 0.1, and 0.1, respectively. Data shuffling was applied to prevent learning bias during the training process. To fit the input dimension requirement, we resized the images to 224 × 224 for transfer learning models. Note that, since our dataset is relatively small compared to what is used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), early stopping was applied during the training process to prevent overfitting, which results in fewer learning iterations.

1) GENERAL CNN
CNN allows us to extract higher representations for the image content [38]. Unlike classical image recognition where the image features are defined by the users, CNN takes the raw pixel data of images, trains the model, then extracts the features automatically for the classification. Since CNN uses partially connected layers and weight sharing, it requires fewer computation resources and makes the learning process more efficient than deep neural network [39].

The CNN model was built using 3 convolutional layers, each of which was followed by a max-pooling layer and Dropout layer to reduce the data dimension and avoid overfitting. The dense layer was taken at the end to perform the classification. L2 regularization technique was applied to improve the model generalization [40]. The training and test results are shown in Fig. 7 (A1 & B1). The training loss decreases from 1.223 to 0.825, and the training accuracy increases from 46.7% to 72.3% as the epoch increases. Meanwhile, the validation loss is stably decreasing from 1.225 to 0.901, and the validation accuracy is increasing from 58.1% to 71.6%. As it can be seen in both figures, the training performance is better than it of the validation at the end, which indicates that the model starts to get overfitting at 7 epochs. Therefore, the model was tested using the weight saved before it gets overfitted. The test loss is 0.912, and the test accuracy is 73.5%. The results show that the CNN model can extract part of the actin cytoskeleton features, and the extracted features are utilized by the neural network for classification. However, its performance is not as good as the proposed graph to vector embedding model. The proposed model has higher abilities in extracting morphology features of the actin cytoskeleton.

2) TRANSFER LEARNING MODELS
Since deep neural network models may take days or even weeks to train on a very large dataset, a way to short-cut this process is to reuse the pre-trained models that were developed for standard benchmark datasets [41], such as the ILSVRC. The best-performing models could be downloaded and reused directly, or integrated into a new model to solve other image recognition problems. Moreover, reusing the pre-trained models helps us capture the inherent data distribution more effectively.

In this study, GoogleNet, Xception, and VGG16 were used to predict the image labels. We excluded the top of the network (i.e., global average pooling layer and the dense output layer), and added our own global average pooling layer based on the output of the base model, followed by a classifier with one unit per class (i.e., three units in total). Since this is a multi-classification problem, softmax was taken as the dense layer’s activation function. The weights of the pre-trained models were frozen in the training process at the beginning of the training. The training and test results are shown in Fig. 7 (B ~ D). For GoogleNet, the model loss decreases from 1.300 and 1.092 to 0.843 and 0.953 for training and validation, respectively, when the epoch increases. The accuracy increases from 53.4% and 59.8% to 67.8% and 66.1% for training and validation, respectively. The test loss is 1.233, and the test accuracy is 59.5%. For Xception, the model loss decreases from 0.949 and 0.841 to 0.655 and 0.750 for training and validation, respectively. Meanwhile, its accuracy increases from 58.1% and 63.6% to 74.0% and 69.3% for training and validation, respectively. The test loss is 0.900, and the test accuracy is 66.8%. For VGG16, its loss decreases from 0.945 and 0.680 to 0.624 and 0.632 for training and validation, respectively. Its accuracy increases from 62.1% and 73.4% to 76.1% and 74.9% for training and validation, respectively. The test loss is 0.710, and the test accuracy is 73.9%. This results clearly show that these three models start overfitting quickly. As we know, the aforementioned models were built to resolve complex image recognition tasks, therefore, dozens of convolutional layers were included in the network architectures that can extract over hundreds of features. Even though transfer learning shows benefits in avoiding overfitting when the target data set is relatively small, in many cases, the model may not always solve the overall problem. Particularly, for each sample image containing the actin cytoskeleton of one single cell, those models were much too powerful. The over presented features lead to quick overfitting during the training process. Even we tried to un freeze the weights of base models, the overfitting still showed up in 4 epochs and resulted in a bad model generalization. This observation indicates that the applied three pre-trained models are not optimal in making predictions on the morphology state of the cellular actin cytoskeleton when compared with the proposed approach.

Although the superiority of the proposed deep learning model has been demonstrated, it can be further improved by increasing the data size during training so that more elaborate cytoskeleton morphological structures can be detected. Moreover, in future work, more advanced techniques, such as the elite opposition-based learning and chaotic k-best gravitational search strategy (EONS) [42] and genetic algorithm (GA)-assisted AdaBoost random forest (GA-ADA-RF)
model [43], could be applied to improve the model’s performance in real-time response, optimization, and adaptability.

V. CONCLUSION
In this paper, a model to recognize the cellular actin cytoskeleton morphology was built using the graph to vector embedding technique together with the neural network in machine learning. The proposed model consists of a skip-gram model followed by a fully connected classifier. The actin cytoskeleton images of NIH/3T3 cells treated by Latrunculin B at different concentrations were taken as the inputs, and the outputs were the classes of the actin cytoskeleton status, i.e., labels defined by treatment concentrations (i.e., the actin depolymerization level). The test accuracy of 80.49% was obtained at the end showing that both structure and quantity features were extracted efficiently from the actin cytoskeleton images. Combine the proposed model with the experimentally quantified cell cytoskeleton and mechanics data from existing studies, mappings between the cytoskeleton classes and cell mechanical properties can be established. Therefore, the proposed model can help to reduce the experiment time and labor significantly as the cellular mechanical property can be directly estimated if the cytoskeleton classification result is closed to that of existing data.

ACKNOWLEDGMENT
The NIH/3T3 cells were provided by Dr. Xuefeng Wang. The authors also thank Ruocheng Yin and Charchit Shukla for helping with the image cropping.

REFERENCES

[1] N. Lichtenstein, B. Geiger, and Z. Kam, “Quantitative analysis of cytochalasin D and latrunculin B on mechanical properties of eukaryotic cells,” J. Mech. Behav. Biomed. Mater., vol. 78, pp. 65–73, Feb. 2018.

[2] K. Mollaiean, Y. Liu, S. Bi, and J. Ren, “Atomic force microscopy study of cytochalasin D and latrunculin B on mechanical properties of eukaryotic cells,” J. Mech. Behav. Biomed. Mater., vol. 78, pp. 65–73, Feb. 2018.

[3] M. Gupta, B. R. Sarangi, J. Deschamps, Y. Nematabakhsh, A. Callan-Jones, F. Margadant, R.-M. Mége, C. T. Lim, R. Voituriez, and B. Ladoux, “Adaptive rheology and ordering of cell cytoskeleton govern matrix rigidity sensing,” Nature Commun., vol. 6, no. 1, p. 7525, 2015.

[4] T. Nakatsuki, B. Schwab, N. C. Thompson, and E. L. Elson, “Distinguishing left and right context for word embeddings,” in Proc. Artif. Intell. Statist., vol. 26, no. 9, pp. 130–138, 2016.

[5] D. Harris and S. Harris, Digital Design and Computer Architecture. San Mateo, CA, USA: Morgan Kaufmann, 2010.

[6] Y. Song, S. Shi, J. Li, and H. Zhang, “Directional skip-graam: Explicitly distinguishing left and right context for word embeddings,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics. Hum. Lang. Technol., vol. 2, 2018, pp. 175–180.

[7] B. Bartunov, D. Kondrashkin, A. Osokin, and D. Vetrov, “Breaking sticks to actin cytoskeleton quantification,” in Proc. IFAC-PapersOnLine, vol. 54, no. 20, pp. 328–333, 2021.

[8] A. E. Maxwell, T. A. Warner, and F. Fang, “Implementation of machine-learning classification in remote sensing: An applied review,” Int. J. Remote Sens., vol. 39, no. 9, pp. 2784–2817, 2018.
Y. Liu, K. Mollaeian, M. H. Shamim, and J. Ren, “Effect of F-actin and microtubules on cellular mechanical behavior studied using atomic force microscope and an image recognition-based cytoskeleton quantification approach,” *Int. J. Mol. Sci.*, vol. 21, no. 2, p. 392, Jan. 2020.

P. Liashchynskyi and P. Liashchynskyi, “Grid search, random search, genetic algorithm: A big comparison for NAS,” 2019, arXiv:1912.06059.

M. Ringnér, “What is principal component analysis?” *Nature Biotechnol.*, vol. 26, no. 3, pp. 303–304, 2008.

D. M. W. Powers, “Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation,” 2020, arXiv:2010.16061.

S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *Proc. Int. Conf. Eng. Technol. (ICET)*, Aug. 2017, pp. 1–6.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

J. Friedman, T. Hastie, and R. Tibshirani, “Regularization paths for generalized linear models via coordinate descent,” *J. Stat. Softw.*, vol. 33, no. 1, p. 1, 2010.

L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, Hershey, PA, USA: IGI Global, 2010, pp. 242–264.

Y. Yuan, X. Mu, X. Shao, J. Ren, Y. Zhao, and Z. Wang, “Optimization of an auto drum fashioned brake using the elite opposition-based learning and chaotic k-best gravitational search strategy based grey wolf optimizer algorithm,” *Appl. Soft Comput.*, vol. 123, Jul. 2022, Art. no. 108947.

J. Ren, Z. Wang, Y. Pang, and Y. Yuan, “Genetic algorithm-assisted an improved AdaBoost double-layer for oil temperature prediction of TRM,” *Adv. Eng. Informat.*, vol. 52, Apr. 2022, Art. no. 101563.

YI LIU received the B.S. degree in mechanical engineering and the M.S. degree from Southwest Jiaotong University, China, in 2013 and 2016, respectively. She is currently pursuing the Ph.D. degree in mechanical engineering and the B.S. degree in computer science with Iowa State University (ISU), IA, USA. Her research interests include modeling, computer vision, and deep learning.

JUNTAO ZHANG received the bachelor’s degree in mechanical engineering from Iowa State University (ISU), IA, USA, in 2020, where he is currently pursuing the Ph.D. degree in mechanical engineering. His research interests include modeling, artificial intelligence, and deep learning.

CHARUKU BHARAT received the B.E. degree (Hons.) in mechanical engineering from BITS Pilani, India, in 2019. He is currently pursuing the master’s degree in mechanical engineering with Iowa State University. His research interest includes learning-based output tracking and control.

JUAN REN (Member, IEEE) received the B.S. degree in mechanical engineering from Xi’an Jiaotong University, China, in 2009, and the Ph.D. degree in mechanical engineering from Rutgers, The State University of New Jersey, in June 2015. She is currently an Associate Professor with the Department of Mechanical Engineering, Iowa State University, where she has been on the Faculty, since August 2015. Her research interests include learning-based output tracking and control, control tools for high-speed scanning probe microscope imaging, mechanotransduction modeling, and nanomechanical measurement and mapping of soft and live biological materials. She received the NSF CAREER Award, in 2018, and currently holds the William and Virginia Binger Professorship at the Department of Mechanical Engineering, ISU. She is the Representative of the IEEE Control Systems Society in the IEEE Nanotechnology Council and an Associate Editor of *Mechatronics* (Elsevier).