A Bayesian regularization-backpropagation neural network model for peeling computations

Saipraneeth Gouravaraju[a], Jyotindra Narayan[a], Roger A. Sauer[a,b,c,d] and Sachin Singh Gautam[a*]

[a] *Department of Mechanical Engineering, Indian Institute of Technology Guwahati, Assam 781039, India*

[b] *Aachen Institute for Advanced Study in Computational Engineering Science (AICES), RWTH Aachen University, Templergraben 55, 52056 Aachen, Germany*

[c] *Department of Mechanical Engineering, Indian Institute of Technology Kanpur, UP 208016, India*

[d] *Faculty of Civil and Environmental Engineering, Gdańsk University of Technology, ul. Narutowicza 11/12, 80-233 Gdańsk, Poland*

Published[†] in *The Journal of Adhesion*, DOI: 10.1080/00218464.2021.2001335
Submitted on 29 June 2021; Revised on 15 October 2021; Accepted on 19 October 2021

Abstract

A Bayesian regularization-backpropagation neural network (BR-BPNN) model is employed to predict some aspects of the gecko spatula peeling, viz. the variation of the maximum normal and tangential pull-off forces and the resultant force angle at detachment with the peeling angle. K-fold cross validation is used to improve the effectiveness of the model. The input data is taken from finite element (FE) peeling results. The neural network is trained with 75% of the FE dataset. The remaining 25% are utilized to predict the peeling behavior. The training performance is evaluated for every change in the number of hidden layer neurons to determine the optimal network structure. The relative error is calculated to draw a clear comparison between predicted and FE results. It is shown that the BR-BPNN model in conjunction with the k-fold technique has significant potential to estimate the peeling behavior.

Keywords: machine learning, adhesion, peeling, artificial neural networks, Bayesian regularization

1 Introduction

The study of peeling is essential in understanding the adhesion characteristics in many applications such as adhesive tapes, micro- and nano-electronics (Komvopoulos, 2003; Zhang et al., 2009), coatings (Sexsmith and Troczynski, 1994), microfiber arrays (Majidi et al., 2006; Schubert et al., 2007), wearable medical bands (Drotlef et al., 2017), and cell adhesion (Zhu, 2000). Peeling problems have been used by many researchers to analyze multiscale adhesion in biological adhesive pads such as in geckos, insects, and spiders (Persson and Gorb, 2003; Sauer, 2009; Labonte and Federle, 2016; Federle and Labonte, 2019), where peeling is an important aspect of detachment.

Peeling, particularly gecko spatula peeling, has been studied extensively using experimental (Autumn et al., 2000; Autumn, 2002), analytical (Tian et al., 2006), and computational methods (Sauer, 2011b; Sauer and Holl, 2013; Gautam and Sauer, 2013, 2014; Agrawal and Gautam,

*corresponding author, email: ssg@iitg.ac.in
[†]This pdf is the personal version of an article whose journal version is available at https://www.tandfonline.com

2013). However, each of these methods comes with its specific limitations. Although experimental methods provide insights into the gecko adhesive system, they are limited in resolution, typically at seta level. To the authors' best knowledge, there have been no experimental studies that explored the adhesive and frictional behaviour at the spatula level owing to the difficulty in isolating a single spatula. Most of the analytical models that study the peeling of gecko spatulae, although they provide insights into the various aspects of the peeling behaviour, they are limited by their inherent assumptions such as steady-state peeling, zero bending stiffness, and linear material response. As such, most of the analytical models are unable to predict the entire peel-off process, including the snap-off behaviour. This necessitates the use of a numerical analysis tool like FEM. However, the computational cost can become very high due to the nonlinear and small scale nature of molecular adhesion as well as the detailed spatula microstructure. The high computational cost can be overcome by reduced models, such as beam models (Sauer and Mergel, 2014), but the cost remains a major limitation of full continuum models. Recently, Gouravaraju et al. (2021a,b) have studied the peeling behaviour of a single gecko spatula. However, as mentioned above, the computational cost of the numerical model is very high. As observed by some authors (Gu et al., 2018; Oishi and Yagawa, 2020; Kim et al., 2020), the use of machine learning techniques such as artificial neural networks has the potential to reduce these computational costs while retaining the accuracy of numerical methods. In particular, Gu et al. (2018) have shown that employing neural networks can significantly reduce the high computational cost of FE simulations. To the best of the authors' knowledge there has been no study that employs machine learning techniques to analyze adhesive peeling and specifically gecko spatula peeling. Therefore, in this work, a Bayesian regularization-based backpropagation neural network (Argatov and Chai, 2019; MacKay, 1992; Burden and Winkler, 2008) is employed to predict the influence of the peeling angle on the peeling force of a gecko spatula. The input data is obtained from the finite element simulations of Gouravaraju et al. (2021a,b), who have used a quasi-continuum finite element model that captures friction due to adhesion at the nanoscale (Sauer and Li, 2007; Mergel et al., 2021).

The remainder of the paper is structured as follows: Section 2 discusses the adhesive friction model and the peeling of the spatula. In section 3 a backpropagation neural network with Bayesian regularization is presented. Section 4 discusses the implementation of the neural network model. Results and discussion are presented in section 5. Finally, section 6 concludes the paper.

2 Peeling using an adhesive friction model

In this section, the adhesive friction model of Mergel et al. (2021) and its application to gecko spatula[‡] peeling by Gouravaraju et al. (2021a,b) are briefly described.

The "Model EA" of Mergel et al. (2021) defines a sliding traction threshold T_s that is non-zero even for tensile normal forces. This sliding threshold depends on the magnitude of the normal traction $T_\mathrm{n} = \|\boldsymbol{T}_\mathrm{n}\|$ due to adhesion between the spatula and the substrate. Further, it is assumed that the interfacial frictional forces act only up to a certain cut-off distance r_c. Then we have,

$$T_\mathrm{s}(r) = \begin{cases} \dfrac{\mu_\mathrm{f}}{J_\mathrm{c}}\Big[T_\mathrm{n}(r) - T_\mathrm{n}(r_\mathrm{c})\Big], & r < r_\mathrm{c}, \\ 0, & r \geq r_\mathrm{c}, \end{cases} \tag{1}$$

[‡]The nanoscale spatulae in geckos are very thin structures (approximately $5 - 10$ nm thick) with a width of around 200 nm that can be modeled effectively as a thin strip (Tian et al., 2006; Pesika et al., 2007; Peng et al., 2010; Sauer, 2011b).

where J_c is the local contact surface stretch (= 1 for rigid substrates), μ_f is the friction coefficient, and r denotes the distance to the substrate surface.

The normal traction $\boldsymbol{T}_n$ is obtained from the variation of the total adhesion potential, which is the summation of individual adhesion potentials acting between the molecules of the substrate and the spatula, and is given as (Sauer and Wriggers, 2009)

$$\boldsymbol{T}_n = \frac{A}{2\pi r_0^3}\left[\frac{1}{45}\left(\frac{r_0}{r}\right)^9 - \frac{1}{3}\left(\frac{r_0}{r}\right)^3\right]\boldsymbol{n}_s\,, \tag{2}$$

where r_0 is the equilibrium distance of the Lennard-Jones potential, A is Hamaker's constant, and $\boldsymbol{n}_s$ is the normal to the substrate.

Similar to Coulomb's friction model, the magnitude of frictional traction $\boldsymbol{T}_f$ is governed by

$$\|\boldsymbol{T}_f\| \begin{cases} < T_s & \text{for sticking,} \\ = T_s & \text{for sliding,} \end{cases} \tag{3}$$

and is computed using a predictor-corrector algorithm (Gouravaraju et al., 2021a). A Neo-Hookean material model is employed to model the spatula response (Bonet and Wood, 2008). For further details on the application of the adhesive friction model, we refer to Gouravaraju et al. (2021a).

The spatula is modeled as a thin two-dimensional strip as shown in Fig. 1. A displacement $\bar{\boldsymbol{u}}$ is applied to the spatula shaft at an angle called the peeling angle θ_p. Nonlinear finite element analysis is employed to solve the resulting mechanical boundary value problem given by the nonlinear equation

$$\mathbf{f}(\mathbf{u}) := \mathbf{f}_{int} + \mathbf{f}_c = \mathbf{0}\,, \tag{4}$$

where $\mathbf{f}_{int}$ and $\mathbf{f}_c$ are the global internal and contact force vectors. The spatula is divided into 240 × 12 finite elements along x and y directions, respectively. To accurately capture the nonlinear contact tractions (see Eqs. (1) and (2)), a local enrichment strategy proposed by Sauer (2011a) is employed. In this strategy, the contact surface is discretized using fourth-order Lagrange polynomials while the bulk is discretized using the standard linear Lagrange polynomials. Plane strain conditions are assumed.

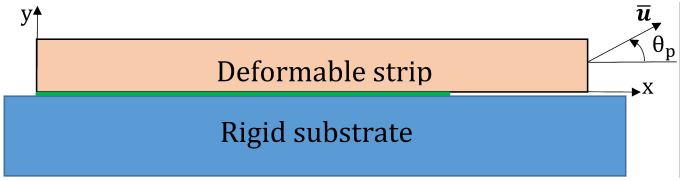

Figure 1: Peeling of a deformable strip from a rigid substrate. The strip is adhering on 75% of the surface.

Although the detailed results of the FE simulation can be found in Gouravaraju et al. (2021a,b), for the sake of completeness, we briefly discuss the peeling process through a representative force-displacement plot. The entire peeling of the spatula can be divided into two phases based on the evolution of the normal and tangential pull-off forces shown in Fig. 2. In the first phase (from displacement $\bar{u}^0$ to $\bar{u}^{max}$), the spatula continuously undergoes stretching due to the fact that it is in a state of partial sliding/sticking near the peeling front. Thus, it accumulates strain energy. At $\bar{u}^{max}$ the spatula is stretched to the maximum as the pull-off forces reach a maximum value. During the second phase (from $\bar{u}^{max}$ to $\bar{u}^{det}$) the spatula fully slides on the substrate. As a result, the spatula relaxes and releases the accumulated energy until it detaches from the substrate spontaneously at $\bar{u}^{det}$. Similar peeling curves are obtained for other peeling angles.

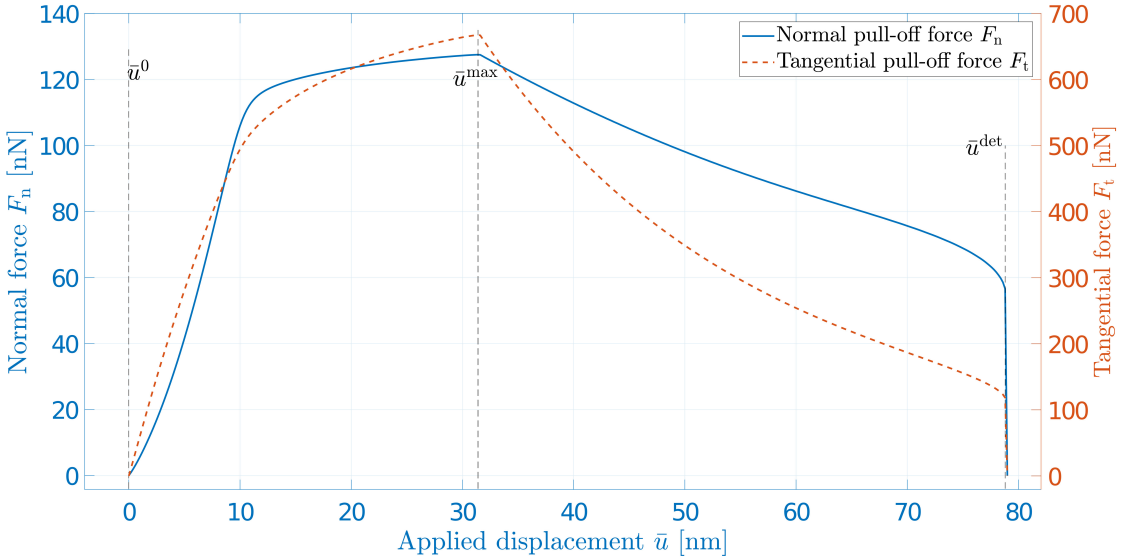

Figure 2: Evolution of normal (F_n) and tangential (F_t) pull-off forces with the applied displacement $\bar{u}$ for peeling angle $\theta_p = 45°$.

In this study, the focus is on three aspects of the peeling process, viz. the maximum normal pull-off force F_n^{max}, the maximum tangential pull-off force F_t^{max}, and the resultant force angle $\alpha = \arctan(F_n/F_t)$ at detachment. It has been shown that depending on the peeling angle θ_p, the maximum pull-off forces F_n^{max} and F_t^{max}, the corresponding displacement $\bar{u}^{max}$ and the detachment displacement $\bar{u}^{det}$ vary considerably (Gouravaraju et al., 2021a). On the other hand, it has been observed (Gouravaraju et al., 2021a,b) that the resultant force angle at detachment α^{det} remains the same irrespective of the peeling angle (see Table B1 in Appendix B).

Remark 1: Note that the geometrical and material parameters are fixed for the case of gecko spatula peeling, see Gouravaraju et al. (2021a,b). Hence, the effect of variation of these parameters is not considered. However, the effect of the geometrical or physical parameters can be incorporated by generating additional FE data and retraining the proposed network with the additional parameters added as input.

3 Bayesian regularization-backpropagation neural network (BR-BPNN)

In this section, a backpropagation neural network (BPNN) along with the Bayesian regularization learning algorithm is described. The background theory on BPNN along with the Bayesian regularization is given in Appendix A. A more detailed discussion can be found in Demuth et al. (2014). BR-BPNN is utilized to achieve better generalization and minimal over-fitting for the trained networks (MacKay, 1992; Burden and Winkler, 2008).

Consider a neural network with training dataset D having n_t input and target vector pairs in the network model, i.e

$$D = \left\{ (\mathbf{u}_1, \mathbf{t}_{o1}), (\mathbf{u}_2, \mathbf{t}_{o2}), \ldots, (\mathbf{u}_{n_t}, \mathbf{t}_{on_t}) \right\}. \tag{5}$$

For each input ($\mathbf{u}$) to the network, the difference between target output ($\mathbf{t}_o$) and predicted output ($\mathbf{a}_o$) is computed as error $\mathbf{e}$. In order to evaluate the performance of the network,

i.e. how well the neural network is fitting the test data, a quantitative measure is needed. This measure is called performance index of the network and is used to optimize the network parameters. The standard performance index $F(\bar{\mathbf{w}})$ is governed by the sum of the squared errors (SSE)

$$F(\bar{\mathbf{w}}) = E_D = \sum_{i=1}^{n_t} (\mathbf{e}_i)^2 = \sum_{i=1}^{n_t} (\mathbf{t}_{oi} - \mathbf{a}_{oi})^{\mathrm{T}} \ (\mathbf{t}_{oi} - \mathbf{a}_{oi}) \ , \tag{6}$$

where $\bar{\mathbf{w}}$ denotes the vector of size K containing all the weights and biases of the network.

In order to generalize the neural network, the performance index of Eq. (6) is modified using a regularization method. A penalty term $(\mu/\nu)E_{\mathrm{w}}$ is added to the performance index $F(\bar{\mathbf{w}})$ (Tikhonov, 1963),

$$F\big(\bar{\mathbf{w}}\big) = \mu \bar{\mathbf{w}}^{\mathrm{T}} \, \bar{\mathbf{w}} + \nu E_D \ = \ \mu E_{\mathrm{w}} + \nu E_D \, , \tag{7}$$

where μ and ν are the regularization parameters and E_{w} represents the sum of the squared network weights (SSW).

Finding the optimum values for μ and ν is a challenging task, as their comparative values set up the basis for the training error. If $\mu \ll \nu$, smaller errors are generated, while if $\mu \gg \nu$, there should be reduced weight size at the cost of network errors (Kayri, 2016). For the purpose of finding the optimum regularization parameters, a Bayesian regularization method is employed.

Considering the network weights $\bar{\mathbf{w}}$ as random variables, the aim is to choose the weights that maximize the posterior probability distribution of the weights $P\big(\bar{\mathbf{w}}|D, \mu, \nu, M_N\big)$ given a certain data D. According to Bayes' rule (MacKay, 1992), the posterior distribution of the weights depends on the likelihood function $P\big(D|\bar{\mathbf{w}}, \nu, M_N\big)$, the prior density $P\big(\bar{\mathbf{w}}|\mu, M_N\big)$, and the normalization factor $P\big(D|\mu, \nu, M_N\big)$ for a particular neural network model M_N and can be evaluated from

$$P\big(\bar{\mathbf{w}}|D, \mu, \nu, M_N\big) = \frac{P\big(D|\bar{\mathbf{w}}, \nu, M_N\big) \, P\big(\bar{\mathbf{w}}|\mu, M_N\big)}{P\big(D|\mu, \nu, M_N\big)} \, . \tag{8}$$

Considering that the noise in the training set has a Gaussian distribution, the likelihood function is given by

$$P\big(D|\bar{\mathbf{w}}, \nu, M_N\big) = \frac{\exp\big(-\nu E_D\big)}{Z_D\big(\nu\big)} \, , \tag{9}$$

where $Z_D = \big(\pi/\nu\big)^{Q/2}$ and $Q = n_t \times N^{n_l}$.

Similarly, assuming a Gaussian distribution for the network weights, the prior probability density $P\big(\bar{\mathbf{w}}|\mu, M_N\big)$ is given as

$$P\big(\bar{\mathbf{w}}|\mu, M_N\big) = \frac{\exp\big(-\mu E_{\mathrm{w}}\big)}{Z_{\mathrm{w}}\big(\mu\big)} \, , \tag{10}$$

where $Z_{\mathrm{w}} = \big(\pi/\alpha\big)^{K/2}$.

The posterior probability with the network weights $\bar{\mathbf{w}}$ can then be expressed as (Kayri, 2016)

$$P\big(\bar{\mathbf{w}}|D, \mu, \nu, M_N\big) = \frac{\exp\big(-\mu E_{\mathrm{w}} - \nu E_D\big)}{Z_F\big(\mu, \nu\big)} = \frac{\exp\big(-F(\bar{\mathbf{w}})\big)}{Z_F\big(\mu, \nu\big)} \, , \tag{11}$$

where $Z_F\big(\mu, \nu\big) = Z_D\big(\nu\big) Z_{\mathrm{w}}\big(\mu\big)$ is the normalization factor.

The complexity of the model M_N is governed by regularization parameters μ and ν, which need to be estimated from the data. Therefore, Bayes' rule is again applied to optimize them from

$$P(\mu,\nu|D,M_N) = \frac{P(D|\mu,\nu,M_N)\,P(\mu,\nu|M_N)}{P(D|M_N)}\,, \tag{12}$$

where $P(\mu,\nu|M_N)$ denotes the assumed uniform prior density for the parameters μ and ν. From Eq. (12), it is evident that maximizing the likelihood function $P(D|\mu,\nu,M_N)$ eventually maximizes the posterior probability $P(\mu,\nu|D,M_N)$. Moreover, it can be noted that the likelihood function in Eq. (12) is the normalization factor of Eq. (8). Therefore, solving for the likelihood function $P(D|\mu,\nu,M_N)$ and expanding the objective function in Eq. (7) around the minimal point $\bar{\mathbf{w}}^*$ via a Taylor series expansion, the optimum values of regularization parameters can be evaluated as follows (Dan Foresee and Hagan, 1997)

$$\mu^* = \frac{\gamma}{2E_{\mathrm{w}}(\bar{\mathbf{w}}^*)} \qquad \text{and} \qquad \nu^* = \frac{Q-\gamma}{2E_D(\bar{\mathbf{w}}^*)}\,, \tag{13}$$

where γ signifies the "number" of effective parameters exhausted in minimizing the error function

$$\gamma = K - \mu^* \mathrm{tr}(\mathbf{H}^*)^{-1}, \text{for} \qquad 0 \le \gamma \le K\,, \tag{14}$$

and $\mathbf{H}^*$ is the Hessian matrix of the objective function evaluated at $\bar{\mathbf{w}}^*$, which is calculated using the Gauss-Newton approximation as (Kayri, 2016)

$$\mathbf{H}^* \approx \mathbf{J}^{\mathrm{T}}\mathbf{J}\,, \tag{15}$$

where $\mathbf{J}$ is the Jacobian matrix formed by the first derivatives of the network errors $\mathbf{e}$ with respect to network weights w_{ij}. In (14), $\mathrm{tr}(\cdot)$ denotes the trace operator. The normalization factor $Z_F(\mu,\nu)$ can then be approximated as (Demuth et al., 2014)

$$Z_F(\mu,\nu) \approx (2\pi)^{K/2}\left(\det(\mathbf{H}^*)\right)^{-1/2}\exp\left(-F(\bar{\mathbf{w}}^*)\right). \tag{16}$$

At the end of the training, a few checks regarding the number of effective parameters are required for better performance of the network (Kayri, 2016). The problem of computing the Hessian matrix at the minimal point $\bar{\mathbf{w}}^*$ is implicitly solved in the Levenberg-Marquardt (LM) training algorithm while finding the minimum of $F(\bar{\mathbf{w}})$. In the LM algorithm, the network weights and biases at the k^{th} iteration are adjusted according to (MacKay, 1992; Dan Foresee and Hagan, 1997)

$$\bar{\mathbf{w}}^{k+1} = \bar{\mathbf{w}}^{k} - \left[\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I}\right]^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{e}\,, \tag{17}$$

where λ denotes the Levenberg's damping factor and $\mathbf{J}^{\mathrm{T}}\mathbf{e}$ is the error gradient, which needs to be close to zero at end of the training.

4 Implementation of BR-BPNN

In this work, the input vector $\mathbf{u}$ of the BR-BPNN models contains seventeen elements with peeling angle values θ_{p} ranging from 10° to 90° at an interval of 5°. The corresponding output vectors are the maximum normal pull-off force $\boldsymbol{F}_{\mathrm{n}}^{\max}$, the maximum tangential pull-off force $\boldsymbol{F}_{\mathrm{t}}^{\max}$, the applied displacement at force maximum $\bar{\boldsymbol{u}}^{\max}$, the resultant force angle at detachment $\boldsymbol{\alpha}^{\det}$, and the applied displacement at detachment $\bar{\boldsymbol{u}}^{\det}$. In general, this input-output dataset is randomly divided into training, validation, and testing sub-datasets. The training dataset

Table 1: K-fold cross-validation with dataset split into five folds. The yellow cells represent testing dataset while the blue cells correspond to training dataset. See Table 2 for explicit details for the folds used in the present work.

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

is used to train the neural network model (which in the current work is carried out using the Bayesian regularization-backpropagation method) and the trained model is further validated with the validation dataset.

As described previously, the neural network model is first trained on the training dataset and its performance is evaluated by making predictions using the testing dataset. However, this type of single-run model-validation method could potentially result in selection-bias, i.e. the accuracy of the model will be highly dependent on the particular choice of the training and testing datasets. In order to assess the effectiveness of a neural network model developed using limited data, as in this work, a cross-validation method called k-fold cross-validation method is employed in the training of neural networks. This helps the neural network to generalize to new or unseen data in a much better manner. In the k-fold cross-validation method the complete dataset is divided into two complementary sub-datasets, i.e. training and testing. In this method, for a given neural network model, the dataset is first randomized and then partitioned (split) in to k almost equal sized sub-datasets called folds. Then, the $k-1$ folds are used to train the neural network. The one remaining fold (i.e., k^{th} fold) is used for testing the performance of the neural network model. This process is repeated k times such that the network is trained and tested on the entire dataset as illustrated in Table 1 which shows the dataset split five times (Split 1 to Split 5) into five folds (Fold 1 to Fold 5). The yellow cells in Table 1 represent testing dataset while the blue cells correspond to training dataset. The performance of the neural network is then reported in terms of the average accuracy obtained from this k-fold cross-validation.

Table 2 give the details of testing dataset and testing dataset used in the k-fold cross-validation, see Appendix B for the FE results. The indices in the table refer to the case number in Table B1 (first column). For each split, the training dataset is used to train the neural network model using Bayesian regularization method and the trained model is further validated with the validation dataset using the fold mentioned in the last column of Table 2. The validation dataset, in other back-propagation training algorithms, is used to optimize the hyperparameters for effective training. The hyperparameters, like the number of neurons in the hidden layer and the learning parameters such as γ and λ, are defined as the variables required for training the neural network. However, for BR-based learning networks, the hyperparameters in the form of the regularization parameters (μ, ν) are implicitly optimized using Eq. (7). Therefore, the validation set is not essentially required in this case for optimizing the network hyperparameters. Finally, the testing dataset is utilized to predict the targeted output $\mathbf{t}_o$ and analyze the model performance, accordingly. Appendix C presents a simple algorithmic overview of the BR-BPNN model developed in the present work.

Table 2: Details of training dataset and testing dataset used in the k-fold cross-validation. The indices refer to the case number (first column) in Table B1.

Split Number	Training dataset indices	Number of training data (N_{train})	Testing dataset indices	Number of testing data (N_{test})	Fold for Testing dataset
Split 1	1, 2, 5, 6, 8, 10, 11, 12, 13, 14, 15, 16, 17	13	3, 4, 7, 9	4	Fold 1
Split 2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 11, 14, 15, 16	14	12, 13, 17	3	Fold 2
Split 3	2, 3, 4, 5, 7, 8, 9, 12, 13, 14, 15, 16, 17	13	1, 6, 10, 11	4	Fold 3
Split 4	1, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17	14	2, 8, 14	3	Fold 4
Split 5	1, 2, 3, 4, 4, 6, 7, 8, 9 10, 11, 12, 13, 14, 17	14	5, 15, 16	3	Fold 5

Next, two BR-BPNN models are formed with different output datasets; the first model has three output vectors and the second model has two output vectors as shown in Tables 3 and 4. The three output vectors for BR-BPNN-I are the applied displacement at force maximum $\bar{\boldsymbol{u}}^{\max}$, the maximum normal pull-off force $\boldsymbol{F}_{\text{n}}^{\max}$, and the maximum tangential pull-off force $\boldsymbol{F}_{\text{t}}^{\max}$. For BR-BPNN-II, the output vectors are the applied displacement at detachment $\bar{\boldsymbol{u}}^{\det}$ and the resultant force angle at detachment $\boldsymbol{\alpha}^{\det}$, respectively. Each output vector consists of $3N_{\text{test}}$ and $2N_{\text{test}}$ elements for models BPNN-I and BPNN-II respectively.

However, only N_{train} elements corresponding to the input training dataset (see Table 2) are selected for training the BPNN models. Then, the input and output vectors are normalized by the corresponding maximum values. The performance of the BR-BPNN models are estimated by comparing the mean square error (MSE) values with the number of neurons in the hidden layer and determining the optimal number. The MSE is computed from the network error E_D in Eq. (6) as

$$\text{MSE} = \frac{1}{n_t} E_D. \tag{18}$$

Table 3: Output dataset for model BR-BPNN-I (see Appendix B for the FE results).

Applied displacement at force maximum	$\bar{\boldsymbol{u}}^{\max} := [\bar{u}_1^{\max}, \bar{u}_2^{\max}, \ldots\ldots, \bar{u}_{16}^{\max}, \bar{u}_{17}^{\max}]^{\text{T}}$
Maximum normal pull-off force	$\boldsymbol{F}_{\text{n}}^{\max} := \left[F_{n_1}^{\max}, F_{n_2}^{\max} \ldots\ldots, F_{n_{16}}^{\max}, F_{n_{17}}^{\max}\right]^{\text{T}}$
Maximum tangential pull-off force	$\boldsymbol{F}_{\text{t}}^{\max} := \left[F_{t_1}^{\max}, F_{t_2}^{\max} \ldots\ldots, F_{t_{16}}^{\max}, F_{t_{17}}^{\max}\right]^{\text{T}}$

Remark 2: Even though the geometric and material parameters for gecko spatula peeling are cosidered fixed, see Remark 1 at the end of Section 2, the proposed model can be extended to predict the influence of these parameters as follows: First an additional FE dataset needs to be generated for each parameter. Then, the input vector of the proposed model needs to be extended to include the additional input parameters. The network can then be retrained to obtain the optimum number of neurons in the hidden layer and the model parameters. The algorithm mentioned in appendix C will then, in principle, work in a similar manner.

Table 4: Output dataset for model BR-BPNN-II (see Appendix B for the FE results).

Applied displacement at detachment	$\bar{\boldsymbol{u}}^{\text{det}} := \left[\bar{u}_1^{\text{det}}, \bar{u}_2^{\text{det}}, \ldots\ldots, \bar{u}_{16}^{\text{det}}, \bar{u}_{17}^{\text{det}}\right]^{\text{T}}$
Resultant force angle at detachment	$\boldsymbol{\alpha}^{\text{det}} := \left[\alpha_1^{\text{det}}, \alpha_2^{\text{det}} \ldots\ldots, \alpha_{16}^{\text{det}}, \alpha_{17}^{\text{det}}\right]^{\text{T}}$

Remark 3: It is worth noting that in this work a neural network-based prediction of adhesion phenomena is proposed rather than applying a curve-fitting-based interpolation technique. At first it may appear that the proposed BR-BPNN models merely interpolate the missing data. However, this is not so due to the following reasons:

- It can be observed from Table B1 that all the output vectors consist of high dimensional data and the change from preceding value to the next one is highly nonlinear. It is well known that for the case of highly nonlinear data, neural networks can provide more flexibility in mapping the input-output relation with accurate tolerances. Moreover, in case more precise results are desired using curve fitting, the selection of high dimensional polynomials increases the computational complexity and eventually the computational time. This is a major drawback of curve fitting.
- Furthermore, even if curve fitting can be used for interpolation, it is pertinent to mention that the generalization capability of the curve fitting technique can not be as accurate as the proposed BR-BPNN when the dimensionality of the data increases. The proposed BR-BPNN models perform well because the procedure to map the input-output dataset is inherently interpreted by the systematic selection of activation function, hyperparameters, neurons, and hidden layer(s). However, in case of curve-fitting, this process is an iterative one left as a user input for the selection of a polynomial function.

Remark 4: It is also worth to mention that the computational time taken for each full finite element run shown in Table B1, depending on the peeling angle value, takes between 15 minutes to 7 hours on a multicore machine with parallel computing. On the other hand the training and the testing of the dataset in Table 2 takes less than a minute for each split on the same machine without any parallel computing option enabled.

5 Results and discussion

This section presents the Bayesian regularization-based backpropagation neural network predictions of the maximum normal pull-off force $F_{\text{n}}^{\max}$, the maximum tangential pull-off force $F_{\text{t}}^{\max}$, and the resultant force angle at detachment α^{det} along with the corresponding displacements $\bar{u}^{\max}$ and $\bar{u}^{\text{det}}$. Predictions of the networks are then compared with the FE results of Gouravaraju et al. (2021a,b) that have not been yet used for training.

To define the optimal structure of each network model, the mean square error (MSE) of Eq. (18) is investigated along with the number of neurons (1 to 10) in the hidden layer. For the two BR-BPNN models (BR-BPNN-I and BR-BPNN-II), training is performed with 1 to 10 hidden neurons. The MSE values for both the models with only one hidden neuron are found to be comparatively high i.e. 7×10^{-3} and 9.045×10^{-4}, being incapable to form an efficient network. However, as the number of hidden neurons increases to two, a major drop in the MSE values (6.793×10^{-4}, and 2.6060×10^{-4}) is recorded. Each model is trained 15 times independently

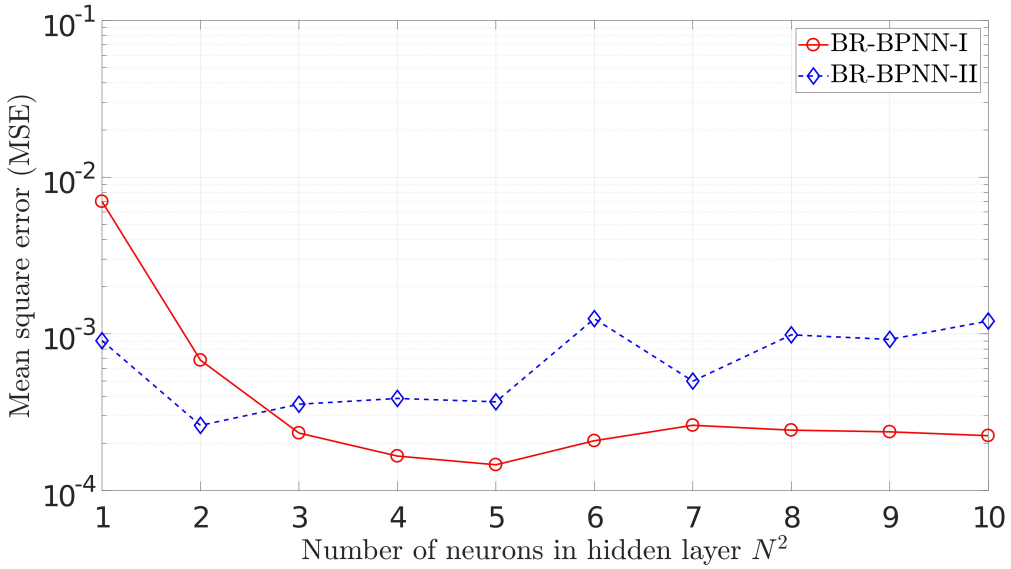

Figure 3: Average mean square error from 5-fold cross-validation with the number of neurons in the hidden layer for different BR-BPNN models.

for different number of neurons to mitigate the unfavorable effects by choosing random initial weights. Each network model is trained for a maximum of 2000 epochs. An epoch is completed when the entire training dataset is passed forward and backward through the network thus updating the weights once. For BPNN-I, the mean square error attains a broad minimum and continuous to decrease between 1 and 5 hidden neurons as shown in Fig. 3. For N^2 greater than 5, the MSE value again starts to rise due to overfitting of the network models. Therefore, for BPNN- I the number of neurons in the hidden layer is selected as 5. The number of neurons in the input and output layers are taken as 1 and 3 as there is one input vector and three output vectors for the BPNN-I model. Following a similar trend, the optimal number of hidden neurons for model BPNN-II is found to be 2, forming the network structure 1-2-2.

Either of the following criteria are selected to terminate or complete the training process: maximum number of epochs reached, minimum value of performance gradient reached, minimum constant value of effective parameters (γ) reached, maximum value of Levenberg's damping factor (λ) attained, or MSE reaching the performance limits. The training results for models BR-BPNN-I and BR-BPNN-II are shown in Tables 5 and 6 respectively. The other network training parameters like the sum of square errors (SSE) (Eq. (6)), sum of square weights (SSW) (E_{w} in Eq. (7)), Levenberg's damping factor, and error gradient (Eq. (17)) values are also shown in Tables 5 and 6.

Table 5: Training parameters for the best configuration (1-5-3) for BR-BPNN-I from 5-fold cross validation.

	Epochs	MSE	SSE(E_D)	SSW (E_W)	Number of effective parameters (γ)	LM Parameter (λ)	Gradient ($\mathbf{J}^{\mathrm{T}}\mathbf{e}$)
Split 1	384	9.79×10^{-4}	3.26×10^{-4}	63.98	22.18	1.0	9.88×10^{-8}
Split 2	300	7.39×10^{-5}	2.47×10^{-5}	80.97	22.38	1.0×10^{10}	1.07×10^{-7}
Split 3	145	2.36×10^{-6}	7.81×10^{-7}	56.61	23.26	1.0×10^{10}	1.31×10^{-7}
Split 4	92	1.86×10^{-6}	6.21×10^{-7}	110.29	24.90	1.0×10^{10}	1.34×10^{-7}
Split 5	106	0.0011	3.62×10^{-4}	46.91	23.15	1.0	9.88×10^{-8}

Table 6: Training parameters for best configuration (1-2-2) for BR-BPNN-II from 5-fold cross validation.

	Epochs	MSE	SSE(E_D)	SSW (E_W)	Number of effective parameters (γ)	LM Parameter (λ)	Gradient ($\mathbf{J}^{\mathrm{T}}\mathbf{e}$)
Split 1	64	1.18×10^{-5}	3.93×10^{-6}	53.71	8.57	1.0×10^{10}	4.23×10^{-7}
Split 2	82	1.27×10^{-5}	4.24×10^{-6}	60.20	8.26	1.0×10^{10}	5.76×10^{-7}
Split 3	43	8.38×10^{-6}	2.79×10^{-6}	37.29	8.31	1.0×10^{10}	7.75×10^{-7}
Split 4	68	5.84×10^{-6}	1.95×10^{-6}	43.26	8.51	1.0×10^{10}	8.93×10^{-7}
Split 5	115	8.94×10^{-6}	2.98×10^{-6}	62.13	8.43	1.0×10^{10}	7.59×10^{-7}

After training the models with input-output datasets with N_{train} datapoints (see Table 2), the testing dataset with N_{test} datapoints (see Table 2) is utilized to predict the corresponding desired output values. The relative error (RE) is used to measure the accuracy of the network predictions. The RE is calculated as the deviation of the predicted result from the desired target result, i.e.

$$\text{RE} = \frac{t_i - a_i}{t_i}\,, \tag{19}$$

where t_i and a_i denote the desired target result and the network prediction for a particular peeling angle of the testing data set, respectively.

5.1 Case I: Maximum normal and tangential pull-off forces

Based on the training parameters from Table 5, Figs. 4, 5, and 6 present the predicted (BR-BPNN-I) results of the maximum normal pull-off force $F_{\text{n}}^{\max}$, maximum tangential pull-off force $F_{\text{t}}^{\max}$ and the corresponding applied displacement $\bar{u}^{\max}$. Since in the present work a 5-fold cross validation method is used the predicted and the desired results across all the splits are shown[§].

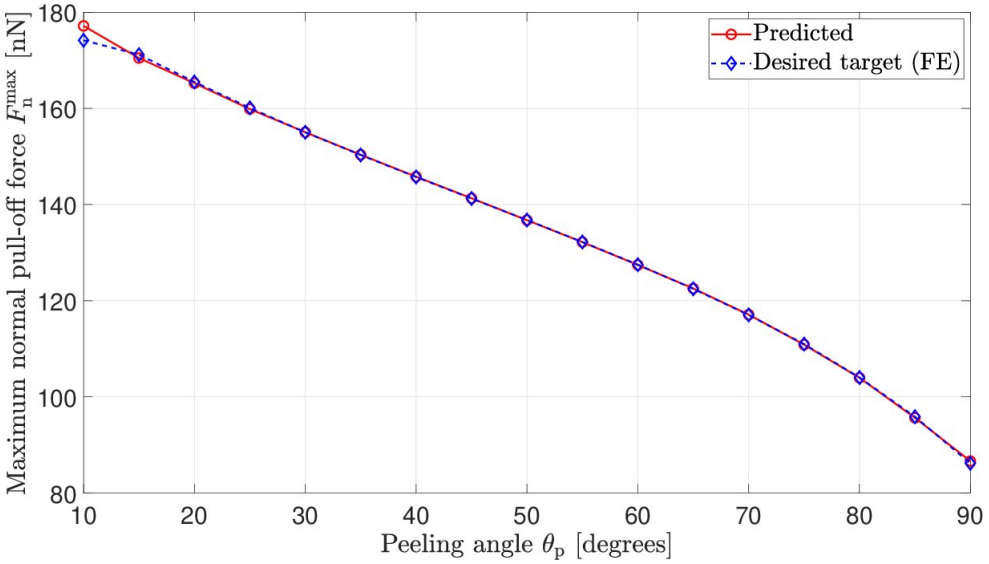

Figure 4: Plot of predicted and desired (FE) value of maximum normal pull-off force $F_{\text{n}}^{\max}$ with the peeling angle θ_{p} across all the splits for model BR-BPNN-I.

[§]The correlation between the split and the predicted indices can be found in Table 2

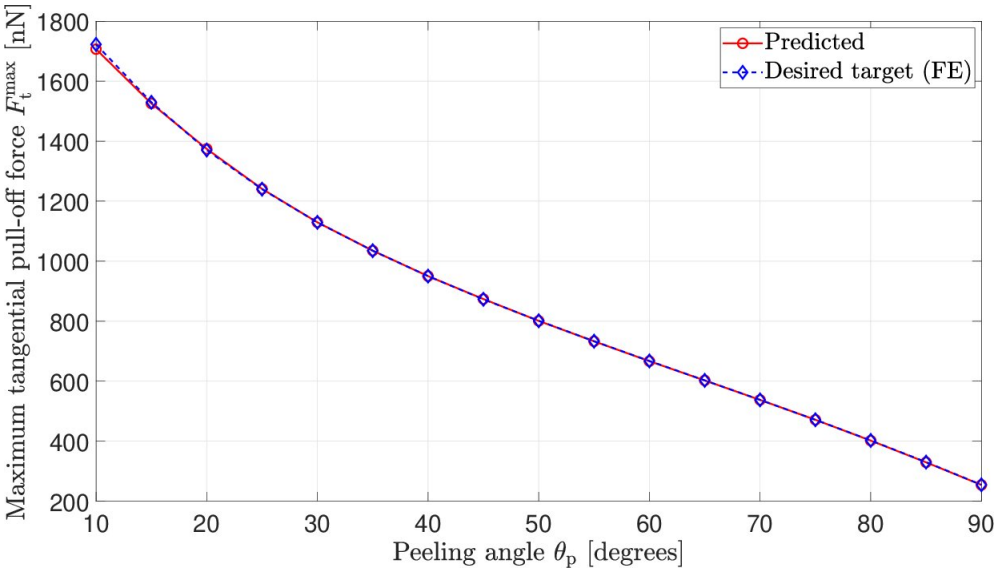

Figure 5: Plot of predicted and desired (FE) value of maximum tangential pull-off force F_t^{max} with the peeling angle θ_p across all the splits for model BR-BPNN-I.

It can be seen from Figs. 4 and 5 that the predicted values of F_n^{max} and F_t^{max} for all angles except $\theta_p = 10°$ are very close to the desired target results (that are obtained by FE). However, for $\theta_p = 10°$, the predicted results show a slightly higher deviation compared to the other tested peeling angles. It is observed from Table 2 that the 10° angle (index 1) is considered as part of the training dataset for the first, second, fourth, and fifth split and the testing dataset for the third split. Although the predicted results for 10° angle are found to be more accurate for the third split, they, however, are computed for all the splits. Furthermore, as can be interpreted from Table 5, the MSE values significantly contribute to those splits which have 10° angle

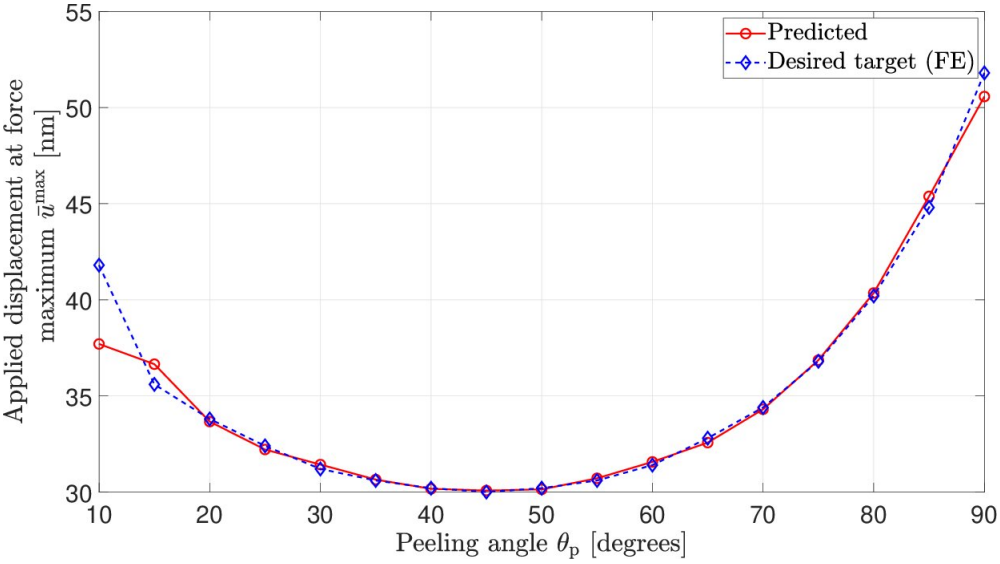

Figure 6: Plot of predicted and desired (FE) value of applied displacement $\bar{u}^{max}$ with the peeling angle θ_p at maximum pull-off force across all the splits for model BR-BPNN-I.

(index 1) in the training dataset. The MSE value in the third split can not compensate the adverse effects of the MSE values in rest of the splits. Therefore, this cumulative effect of the MSE values has become instrumental in creating a disparity between the predicted results from the testing dataset and the desired results from the FE model. In the case with 20° angle, the disparity is substantially reduced by compensating the adverse effects of the training splits

Table 7: Relative error (RE) for the predictions of model BR-BPNN-I.

	$F_{\mathrm{n}}^{\max}$	$F_{\mathrm{t}}^{\max}$	$\overline{u}^{\max}$
Maximum RE (%)	1.78	0.91	9.68
Minimum RE (%)	2.04×10^{-4}	0.0076	0.06
Average RE (%)	0.24	0.19	1.22

(first, second, third, and fifth) through the testing split (i.e., the fourth split) which has the smallest MSE value. The predictions are a little different for $\bar{u}^{\max}$ as shown in Fig. 6 where significant differences are found for $\theta_{\mathrm{p}} = 10°$, $20°$, and $90°$. This can also be observed from Table 7, which lists the relative error (RE) for the all the tested peeling angles. From the table it can be seen that the maximum relative error for the case of displacement $\overline{u}^{\max}$ is 9.68% while the average relative error is around 1.22%. The average relative error for the case of maximum normal and tangential forces is found to be very small.

5.2 Case II: Resultant force angle at detachment

Figures 7 and 8 show the predictions for the output dataset of BR-BPNN-II, i.e. the applied displacement at detachment $\bar{u}^{\mathrm{det}}$ and the resultant force angle at detachment α^{det} using the corresponding training parameters from Table 6. Again, as mentioned previously, since in the present work a 5-fold cross validation method is used, the predicted and the desired results across all the splits are shown.

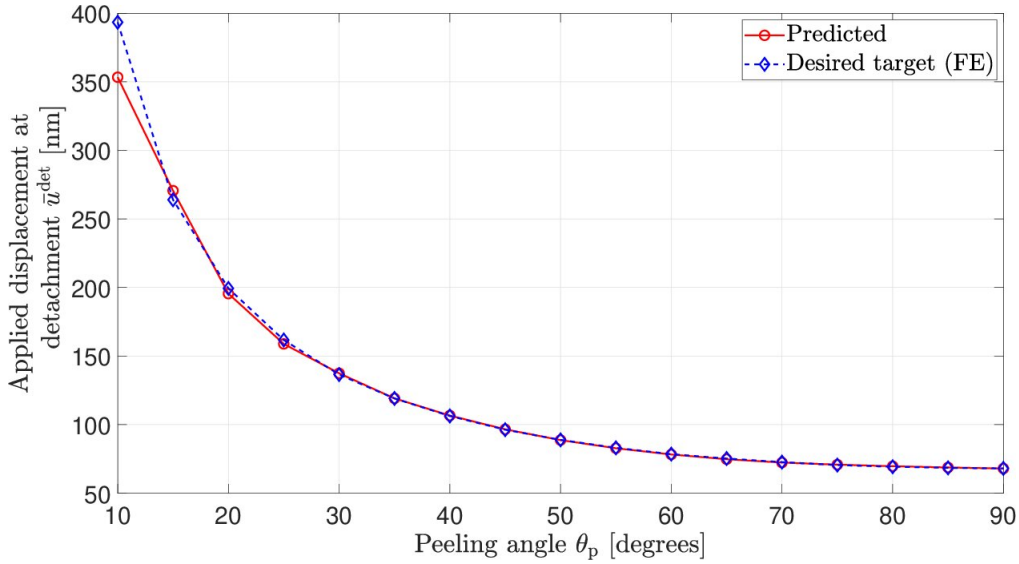

Figure 7: Plot of predicted and desired (FE) value of applied displacement at detachment $\bar{u}^{\mathrm{det}}$ with the peeling angle θ_{p} across all the splits for model BR-BPNN-II.

It can be seen from Fig. 7 that the predicted values of $\overline{u}^{\mathrm{det}}$ for all the angles except for $\theta_{\mathrm{p}} = 10°$ are very close to the desired target FE results. The maximum, minimum and the average RE values, given in Table 8, are estimated to be $9.85\%, 0.15\%$, and 1.24%, respectively. Although the predicted results are evaluated for all the splits, the most effective results are observed for the third split for the case of $\theta_{\mathrm{p}} = 10°$. Similar to the reasoning in Section 5.1, the pestilential effects of cumulative MSE values pertaining to the first, second, fourth, and the fifth split cannot be compensated by the MSE value in the third split. This results in discrepancies between the neural network predicted outputs and FE-based desired outputs. However, in case of the $\theta_{\mathrm{p}} = 20°$ angle, this problem is addressed by mitigating the adverse effects of the training splits

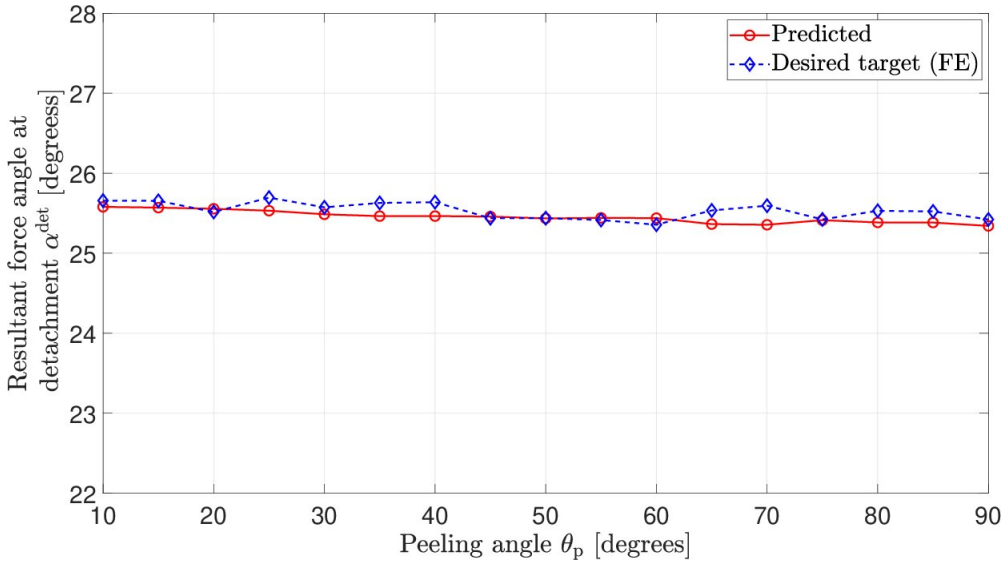

Figure 8: Plot of predicted and desired (FE) value of resultant force angle at detachment $\boldsymbol{\alpha}^{\text{det}}$ with the peeling angle θ_{p} across all the splits for model BR-BPNN-II.

(first, second, third, and fifth) with the benefits of the smallest MSE value in the testing split (i.e., the fourth split). As shown in Fig. 8, the predicted values of α^{det} are also very close to the desired target FE results. The maximum, minimum and the average RE values corresponding to the α^{det} predictions are estimated to be $0.66\%, 0.06\%$, and 0.30%, respectively. It can be observed that the predictions are very accurate even outside of the training data set.

Table 8: Relative error (RE) for the predictions of model BR-BPNN-III.

	α^{det}	$\bar{u}^{\text{det}}$
Maximum RE (%)	0.66	9.85
Minimum RE (%)	0.06	0.15
Average RE (%)	0.30	1.24

From all these results, it can be observed that for both the BR-BPNN models, the predictions are very close to the target outputs value except for $\theta_{\text{p}} = 10°$. Further, for both the BR-BPNN-I and BR-BPNN-II models, the deviations in the predictions are larger for displacements rather than forces. Whereas in case of BR- BPNN-I and BR-BPNN-II, $\bar{u}^{\max}$, and $\bar{u}^{\text{det}}$ vary quite abruptly near $\theta_{\text{p}} = 10°$. This is because for both BR-BPNN-I and BR-BPNN-II , $\bar{u}^{\max}$ and $\bar{u}^{\text{det}}$ vary quite abruptly at $\theta_{\text{p}} = 10°$ (as seen in Figures 6 and 7) and thus can be considered as outliers.

The important advantage of these ANN models lies in the significant reduction in computational cost. It is observed that the time to train the networks with the data corresponding to all the testing peeling angles of each split for both networks is hardly more than one minute. Similarly, once the network is trained, any number of predictions can be made within minutes. Thus, using FE models in conjunction with ANNs has the potential to significantly reduce the computational time leading to faster analysis once the required data has been obtained. This gives a particularly big advantage when the data is obtained using experiments.

6 Conclusions

An artificial neural network model is constructed in the present work to study the peeling behavior of a thin strip such as a gecko spatula. In particular, the variation of the maximum normal and tangential pull-off forces, the corresponding applied displacement, the resultant force angle and the applied displacement at detachment as a function of the peeling angle are investigated. The input data is obtained from the finite element analysis of Gouravaraju et al. (2021a,b). Bayesian regularization in conjunction with k-fold cross validation method is used to form two separate networks. The two networks correspond to (a) the maximum normal and tangential pull-off force and the corresponding applied displacement, and (b) the resultant force angle and the applied displacement at detachment. The number of hidden neurons in each model are evaluated based on their respective mean square errors. From all the results, the maximum and minimum relative deviations of the predicted values from the FE results are found to be 9.85% and 0.0076% respectively. Based on the results, it can be concluded that the Bayesian regularization-based backpropagation neural networks can be employed to successfully study peeling problems. The present work successfully shows that utilizing ANN algorithms can significantly reduce the computational time. Further, the proposed neural network models can be extended to predict the influence of various geometrical, material, and environmental factors on gecko spatula peeling. Another interesting problem that can be investigated using BR-BPNN is the constitutive modeling for the hierarchical structures in the gecko adhesion mechanism.

Acknowledgments

The authors gratefully acknowledge the support from SERB, DST, under projects SB/FTP/ETA-0008/2014 and IMP/2019/000276.

A Background theory on BPNN with Bayesian regularization

A classical neural network architecture mimics the function of the human brain. The brain neurons and their connections with each other form an equivalence relation with neural network neurons and their associated weight values (w). In a single layer network with multiple neurons, each element u_j of an input vector is associated with each neuron i with a corresponding weight w_{ij}. A constant scalar term called bias b_i corresponding to each neuron, which is like a weight, is generally introduced in order to increase the flexibility of the network. This bias b_i is multiplied by a scalar input value (chosen to be 1 here) and is added to the weighted sum $\sum_j w_{ij}u_j$ of the vector components u_j to form a net input n_i. This net input n_i is then passed to an activation function f (also called transfer function) that produces an output value a_i. In general, a neural network consists of two or more layers. Adding a hidden layer of neurons between the input layer and output layer constitutes a multi-layer neural network, also named shallow neural network. The addition of more than one hidden layer in the multi-layer neural network is called a deep neural network.

Traditionally, a BPNN model, a kind of multi-layer neural network, comprises three layers: an input layer, one or more hidden layers, and an output layer, as shown in Fig. A1. The input layer associates the input vector $\mathbf{u}$ having R elements with input weight matrix $\mathbf{W}^1$ and first bias vector $\mathbf{b}^1$ to yield an effective input $\mathbf{n}^1$ to the activation function $\mathbf{f}^1$, which produces an output vector $\mathbf{a}^1$. The output vector $\mathbf{a}^1$ from the first layer forms the input to the hidden layer

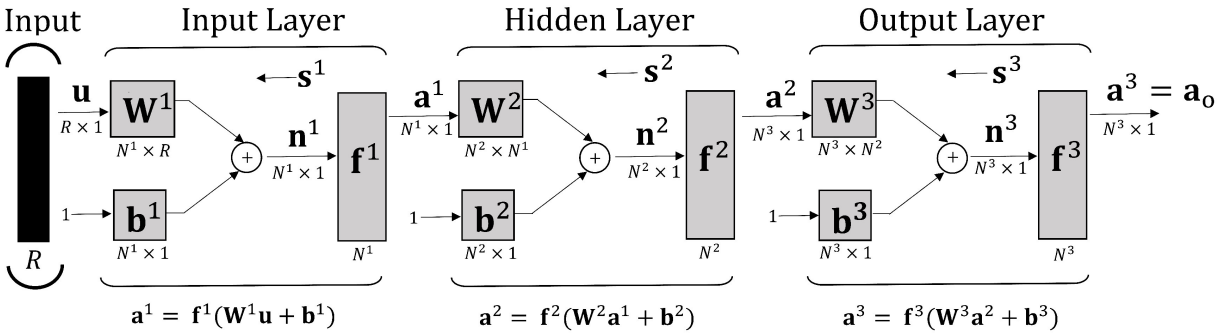

Figure A1: A typical backpropagation neural network with input, hidden, and output layers. Adapted from Demuth et al. (2014).

and is associated with the weight matrix $\mathbf{W}^2$ and bias vector $\mathbf{b}^2$ of the hidden layer. At last, the hidden layer output $\mathbf{a}^2$ is given as an input to the output layer and delivers a predicted output $\mathbf{a}^3$ with weight matrix $\mathbf{W}^3$ and bias vector $\mathbf{b}^3$. In a neural network with a total of n_l number of layers, the weight matrix $\mathbf{W}^l$ and bias vector $\mathbf{b}^l$ for layer l (where $l = 1, 2, \ldots, n_l$) can be written as

$$\mathbf{W}^l = \begin{bmatrix} w^l_{11} & w^l_{12} & w^l_{13} & \ldots & w^l_{1R} \\ w^l_{21} & w^l_{22} & w^l_{23} & \ldots & w^l_{2R} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w^l_{N^l 1} & w^l_{N^l 2} & w^l_{N^l 3} & \ldots & w^l_{N^l R} \end{bmatrix}, \qquad \mathbf{b}^l = \begin{bmatrix} b^l_1 \\ b^l_2 \\ \vdots \\ b^l_{N^l} \end{bmatrix}, \tag{20}$$

where N^l denotes the number of neurons in layer l and the effective input $\mathbf{n}^l$ is then given as

$$\mathbf{n}^l = \mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l, \qquad \text{with} \quad \mathbf{a}^0 = \mathbf{u}. \tag{21}$$

The number of neurons in the input layer (N^1) and output layer (N^3) is linked to the number of input and output vectors, respectively. However, the number of neurons in the hidden layer (N^2) are accountable for the quantification of the weights and biases. The optimal network structure is versed by the optimum number of neurons in each layer required for the training and denoted as N^1-N^2-N^3. A variety of activation functions are used in backpropagation neural network, viz., hard limit, linear, sigmoid, log-sigmoid, hyperbolic tangent sigmoid (Demuth et al., 2014). In the current work, linear activation functions are employed in all the layers according to which, the output is equal to the input i.e. $\mathbf{a}^l = \mathbf{n}^l$.

The network error $\mathbf{e}$ is calculated by subtracting predicted output $\mathbf{a}_o$ from target output $\mathbf{t}_o$. The sensitivity $\mathbf{s}$, which measures how the output of the network changes due to perturbations in the input, is back-propagated from output layer ($\mathbf{s}^3$) to input layer ($\mathbf{s}^1$) via the hidden layer ($\mathbf{s}^2$). Through the backpropagation process, the error of the neurons in the hidden layer is estimated as the backward weighted sum of the sensitivity. Thereafter, to update weights, different learning algorithms are used in association with the sensitivity such as the steepest descent, LM, and conjugate gradient algorithms. The sensitivity at layer l is calculated using the recurrence relation (Demuth et al., 2014)

$$\mathbf{s}^l = \dot{\mathbf{F}}^l\left(\mathbf{n}^l\right) \mathbf{W}^{l+1} \mathbf{s}^{l+1}, \quad \text{where } l = n_l - 1, \ldots, 2, 1, \tag{22}$$

$$\text{with} \quad \mathbf{s}^{n_l} = \dot{\mathbf{F}}^{n_l}\left(\mathbf{n}^{n_l}\right) \left(\mathbf{t}_o - \mathbf{a}_o\right), \tag{23}$$

where $\dot{\mathbf{F}}^l(\mathbf{n}^l)$ is a diagonal matrix containing the partial derivatives of the activation function $\mathbf{f}^l$ with respect to the net inputs $\mathbf{n}^l$ and is given as

$$\dot{\mathbf{F}}^l(\mathbf{n}^l) = \begin{bmatrix} \dot{f}^l(n_1^l) & 0 & \dots & 0 \\ 0 & \dot{f}^l(n_2^l) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \dot{f}^l(n_{N^l}^l) \end{bmatrix}, \qquad \text{where} \quad \dot{f}^l\left(n_j^l\right) = \frac{\partial f^l\left(n_j^l\right)}{\partial n_j^l}, \tag{24}$$

and for the considered linear activation function is equal to the identity matrix.

The purpose of a backpropagation neural network model is to ensure a network with small deviations for the training dataset and supervise the unknown inputs effectively. The intricacy of the BPNN, monitored by neurons in the hidden layer and their associated weights, leads to overfitting, i.e. the network tries to make the error as small as possible for the training set but performs poorly when new data is presented. However, a robust network model should be able to generalize well, i.e. it should predict well even when presented with new data. Therefore, Bayesian regularization based learning of BPNN models is utilized to achieve better generalization and minimal over-fitting for the trained networks (MacKay, 1992; Burden and Winkler, 2008).

B Results from finite element simulations

Table B1 lists the values of the maximum normal force $F_{\mathrm{n}}^{\max}$, maximum tangential force $F_{\mathrm{t}}^{\max}$, applied displacement at force maximum $\bar{u}^{\max}$, applied displacement at $\bar{u}^{\det}$, and resultant force angle at detachment $\alpha^{\det}$ for different peeling angles as obtained by Gouravaraju et al. (2021a,b) using nonlinear finite element analysis.

Table B1: Data from finite element results of Gouravaraju et al. (2021a,b).

Case	Peeling angle θ_{p} [degrees]	Applied displacement at force max. $\bar{u}^{\max}$ [nm]	Max. normal pull-off force $F_{\mathrm{n}}^{\max}$ [nN]	Max. tangential pull-off force $F_{\mathrm{t}}^{\max}$ [nN]	Applied displacement at at detachment $\bar{u}^{\det}$ [nm]	Resultant force angle at detachment $\alpha^{\det}$ [degrees]
1	10	41.8	174.1584	1722.719	393.4	25.64973
2	15	35.6	171.1613	1529.699	263.8	25.57726
3	20	33.8	165.5169	1370.545	199.6	25.56427
4	25	32.4	160.1255	1240.153	161.6	25.59890
5	30	31.2	155.0284	1129.391	136.6	25.60988
6	35	30.6	150.3356	1034.944	119.0	25.55115
7	40	30.2	145.7655	950.3074	106.2	25.55958
8	45	30.0	141.2537	872.9422	96.6	25.61840
9	50	30.2	136.8172	801.4117	89.2	25.65779
10	55	30.6	132.2554	733.2346	83.4	25.62680
11	60	31.4	127.5051	667.3803	78.8	25.53845
12	65	32.8	122.5176	602.7001	75.4	25.66338
13	70	34.4	117.0706	537.6730	72.6	25.51802
14	75	36.8	110.9777	471.2534	70.6	25.49363
15	80	40.2	104.0514	402.4569	69.4	25.69447
16	85	44.8	95.87533	330.1474	68.4	25.44845
17	90	51.8	86.18540	254.5306	68.2	25.49894

C Framework of Bayesian regularization-based backpropagation

The algorithm for the Bayesian regularization based backpropagation is composed of the following steps:

1. Pick training data set D containing the N_{train} cases specified in Tables 2, 3 and 4, and Appendix B.
   (a) Input vector, $\mathbf{u}$: Peeling angles θ_{p}
   (b) Target output vector, $\mathbf{t}_{\text{o}}$: $\bar{\boldsymbol{u}}^{\max}$, $\boldsymbol{F}_{\text{n}}^{\max}$, $\boldsymbol{F}_{\text{t}}^{\max}$ (for BR-BPNN-I)
   $\bar{\boldsymbol{u}}^{\det}$, $\boldsymbol{\alpha}^{\det}$ (for BR-BPNN-II)
2. Initialize neural network with
   (a) Number of neurons in the input layer equal to the number of input vectors, which is equal to 1 for both the BR-BPNN models as described in step 1(a), i.e. $N^1 = 1$.
   (b) Number of neurons in the output layer equal to the number of output vectors, which is equal to $N^3 = 3$ for model BR-BPNN I and $N^3 = 2$ for model BR-BPNN II, respectively, as described in Tables 3 and 4.
   (c) Number of neurons in the hidden layer equal to one, i.e. $N^2 = 1$.
3. Set learning method to Bayesian regularization
   (a) Set maximum number of epochs to 2000.
   (b) Divide the training data set as per Table 2 using k-fold cross validation.
4. Train the network
   (a) Compute regularization parameters μ and ν using Eq. (13).
   (b) Backpropagate sensitivities calculated using Eqs. (22) and (23).
   (c) Update weights using Eq. (17).
5. Compute mean square error (MSE) using Eq. (18).
6. Loop over steps 4 and 5 with different number of neurons in the hidden layer.
7. Plot the MSE with number of neurons in the hidden layer as in Fig. 3.
8. Select the number of neurons in the hidden layer to be the value from which MSE attains a broad minimum and decreases as N^2 is further increased. This determines the optimal network structure N^1-N^2-N^3.
9. Retrain the neural network model with optimal network structure from step 8.
10. Save the model parameters (using Tables 5 and 6) along with weights and biases.
11. Using the saved parameters in step 10, predict for the testing dataset as in Tables 2, 3 and 4.

References

Agrawal, V. and Gautam, S. S. (2013). NURBS based isogeometric analysis for stable and accurate peeling computations. *Sadhana*, 46:3.

Argatov, I. I. and Chai, Y. S. (2019). An artificial neural network supported regression model for wear rate. *Tribol. Int.*, 138:211–214.

Autumn, K. (2002). Mechanisms of Adhesion in Geckos. *Integr. Comp. Biol.*, 42(6):1081–1090.

Autumn, K., Liang, Y. A., Hsieh, S. T., Zesch, W., Chan, W. P., Kenny, T. W., Fearing, R., and Full, R. J. (2000). Adhesive force of a single gecko foot-hair. *Nature*, 405(6787):681–685.

Bonet, J. and Wood, R. D. (2008). *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, London, 2nd edition.

Burden, F. and Winkler, D. (2008). Bayesian Regularization of Neural Networks. In *Methods Mol. Biol.*, pages 23–42. Humana Press.

Dan Foresee, F. and Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian learning. In *Proc. Int. Conf. Neural Networks*, volume 3, pages 1930–1935. IEEE.

Demuth, H. B., Beale, M. H., De Jess, O., and Hagan, M. T. (2014). *Neural network design*. Martin Hagan, 2nd edition.

Drotlef, D., Amjadi, M., Yunusa, M., and Sitti, M. (2017). Bioinspired Composite Microfibers for Skin Adhesion and Signal Amplification of Wearable Sensors. *Adv. Mater.*, 29(28):1701353.

Federle, W. and Labonte, D. (2019). Dynamic biological adhesion: mechanisms for controlling attachment during locomotion. *Philos. Trans. R. Soc. B Biol. Sci.*, 374(1784):20190199.

Gautam, S. S. and Sauer, R. A. (2013). An energy-momentum-conserving temporal discretization scheme for adhesive contact problems. *Int. J. Num. Meth. Engrg.*, 93(10):1057–1081.

Gautam, S. S. and Sauer, R. A. (2014). A composite time integration scheme for dynamic adhesion and its application to gecko spatula peeling. *Int. J. Comp. Methods*, 11(05):1350104 (1–28).

Gouravaraju, S., Sauer, R. A., and Gautam, S. S. (2021a). Investigating the normal and tangential peeling behaviour of gecko spatulae using a coupled adhesion-friction model. *J. Adhes.*, 97(10):952–983.

Gouravaraju, S., Sauer, R. A., and Gautam, S. S. (2021b). On the presence of a critical detachment angle in gecko spatula peeling - a numerical investigation using an adhesive friction model. *J. Adhes.*, 97:1234–1254.

Gu, G. X., Chen, C.-T., Richmond, D. J., and Buehler, M. J. (2018). Bioinspired hierarchical composite design using machine learning: simulation, additive manufacturing, and experiment. *Mater. Horiz.*, 5:939–945.

Kayri, M. (2016). Predictive Abilities of Bayesian Regularization and Levenberg–Marquardt Algorithms in Artificial Neural Networks: A Comparative Empirical Study on Social Data. *Math. Comput. Appl.*, 21(2):20.

Kim, Y., Yang, C., Kim, Y., Gu, G. X., and Ryu, S. (2020). Designing an adhesive pillar shape with deep learning-based optimization. *ACS Appl. Mater. Interf.*, 12(21):24458–24465.

Komvopoulos, K. (2003). Adhesion and friction forces in microelectromechanical systems: mechanisms, measurement, surface modification techniques, and adhesion theory. *J. Adhes. Sci. Technol.*, 17(4):477–517.

Labonte, D. and Federle, W. (2016). Biomechanics of shear-sensitive adhesion in climbing animals: peeling, pre-tension and sliding-induced changes in interface strength. *J. R. Soc. Interface*, 13(122):20160373.

MacKay, D. J. C. (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Comput.*, 4(3):448–472.

Majidi, C., Groff, R. E., Maeno, Y., Schubert, B., Baek, S., Bush, B., Maboudian, R., Gravish, N., Wilkinson, M., Autumn, K., and Fearing, R. S. (2006). High Friction from a Stiff Polymer Using Microfiber Arrays. *Phys. Rev. Lett.*, 97(7):076103.

Mergel, J. C., , Scheibert, J., and Sauer, R. A. (2021). Contact with coupled adhesion and friction: computational framework, applications, and new insights. *J. Mech. Phys. Solids*, 146:104194.

Oishi, A. and Yagawa, G. (2020). A surface-to-surface contact search method enhanced by deep learning. *Comput. Mech.*, 65(4):1125–1147.

Peng, Z. L., Chen, S. H., and Soh, A. K. (2010). Peeling behavior of a bio-inspired nano-film on a substrate. *Int. J. Solids Struct.*, 47(14-15):1952–1960.

Persson, B. N. J. and Gorb, S. (2003). The effect of surface roughness on the adhesion of elastic plates with application to biological systems. *J. Chem. Phys.*, 119(21):11437–11444.

Pesika, N. S., Tian, Y., Zhao, B., Rosenberg, K., Zeng, H., McGuiggan, P., Autumn, K., and Israelachvili, J. N. (2007). Peel-Zone Model of Tape Peeling Based on the Gecko Adhesive System. *J. Adhes.*, 83(4):383–401.

Sauer, R. A. (2009). Multiscale modelling and simulation of the deformation and adhesion of a single gecko seta. *Comput. Methods Biomech. Biomed. Engin.*, 12(6):627–640.

Sauer, R. A. (2011a). Enriched contact finite elements for stable peeling computations. *Int. J. Numer. Methods Eng.*, 87(6):593–616.

Sauer, R. A. (2011b). The Peeling Behavior of Thin Films with Finite Bending Stiffness and the Implications on Gecko Adhesion. *J. Adhes.*, 87(7-8):624–643.

Sauer, R. A. and Holl, M. (2013). A detailed 3D finite element analysis of the peeling behaviour of a gecko spatula. *Comput. Methods Biomech. Biomed. Engin.*, 16(6):577–591.

Sauer, R. A. and Li, S. (2007). A contact mechanics model for quasi-continua. *Int. J. Numer. Methods Eng.*, 71(8):931–962.

Sauer, R. A. and Mergel, J. C. (2014). A geometrically exact finite beam element formulation for thin film adhesion and debonding. *Finite Elem. Anal. Des.*, 86:120–135.

Sauer, R. A. and Wriggers, P. (2009). Formulation and analysis of a three-dimensional finite element implementation for adhesive contact at the nanoscale. *Comput. Methods Appl. Mech. Eng.*, 198(49-52):3871–3883.

Schubert, B., Majidi, C., Groff, R. E., Baek, S., Bush, B., Maboudian, R., and Fearing, R. S. (2007). Towards friction and adhesion from high modulus microfiber arrays. *J. Adhes. Sci. Technol.*, 21(12-13):1297–1315.

Sexsmith, M. and Troczynski, T. (1994). Peel adhesion test for thermal spray coatings. *J. Therm. Spray Technol.*, 3(4):404–411.

Tian, Y., Pesika, N. S., Zeng, H., Rosenberg, K., Zhao, B., McGuiggan, P., Autumn, K., and Israelachvili, J. N. (2006). Adhesion and friction in gecko toe attachment and detachment. *Proc. Natl. Acad. Sci.*, 103(51):19320–19325.

Tikhonov, A. N. (1963). Solution of ill-posed problems and the regularization method. *Dokl. Akad. Nauk SSSR*, 151:501–504.

Zhang, X., Liu, Y., Liu, Y., and Ahmed, S. I. U. (2009). Controllable and switchable capillary adhesion mechanism for bio-adhesive pads: Effect of micro patterns. *Sci. Bull.*, 54(10):1648–1654.

Zhu, C. (2000). Kinetics and mechanics of cell adhesion. *J. Biomech.*, 33(1):23–33.