Feasibility Investigation of Machine Learning for Electronic Reliability Analysis using FEA

Qi yichen, Matrikelnr.: 4905201

Day of submission: 26.10.2022

Abstract

The purpose of this paper is to propose a machine learning framework for predicting the reliability of solder joints using finite element analysis(FEA). The model predicts the equivalent elastic strain on the solder joint based on temperature, amplitude, and geometry of the PCB, chip, and solder joint. The results show that the prediction model has a high accuracy in the shortest possible time. Since the data sets used in this paper are all from FEA, if the model trained from these data is used to predict the data in the experiments, some common measurement errors in the experiments and the industrial manufacturing tolerance of chips and PCBs will lead to inaccurate prediction results. Therefore, some artificial Gaussian noise is added to the ML model to strengthen the robustness of the ML model.

Keywords: machine learning; FEA; artificial Gaussian noise; solder joint reliability prediction

Contents

1 Introduction

In recent years, the large-scale development and widespread deployment of electronic devices and sophisticated electronic systems have transformed the use of high technology in modern society. So the safe and highly reliable utilization of electronic devices within the specified working hours is undoubtedly one of the most pressing issues facing every manufacturer and researcher.

Among many unresolved issues, reliability assessment and life prediction of solder joints in electronic devices have been a long-standing problem. Electronic reliability is a measure of the frequency of equipment failures as a function of time. It has a major impact on maintenance and repair costs and the continuity of service. And solder joint reliability is one of the most important factors in electronic reliability analysis. The role of the solder joint is to create an electrical connection between electronic devices(like a chip) and printed circuit boards (PCB). Therefore, stresses and deformations in solder joints are critical to electronic reliability analysis. Among several external factors that affect the reliability of solder joints, thermal cycling(T.C.)[14] and vibration[6] are the main ones that mainly involve each electronic system. In addition, some internal factors such as the properties of different solder joints, geometry size of PCBs, chips, and solder joints also affect the reliability of the solder joint to some extent. In thermal cycling, different materials have different CTEs, and when they are next to each other in a heterogeneous device, the CTE mismatch between them leads to deformation and stress/strain, resulting in cracked solder joints.

The simulations in this paper test the maximum equivalent elastic strain value generated in a solder joint during cyclic vibration under isothermal conditions, which means that the temperature is constant in one simulation and no cracking of the solder joint due to CTE mismatch in thermal cycling occurs. In the experiment, electronic devices may be subjected to many different forms of vibration over a wide range of frequencies and acceleration levels[6]. For vibration analysis, the vibration mode and the intrinsic frequency of the vibrating body must be determined. Here we choose the first harmonic mode because it has the largest displacement amplitude and has the largest displacement-induced stress[3]. Vibration-induced stresses usually lead to fatigue failure of electronic components. Vibratory fatigue failure of solder joints is typically evaluated for reliability using a high cycle fatigue life model, represented by an S - N (stress-life) curve[11]. Different temperatures also affect the stress and deformation to which the solder joint is subjected. For example, high temperatures make the material more flexible and subjected to more stress, while low temperatures make the material more stubborn and subjected to less stress.

In order to accurately predict solder joint life, appropriate intrinsic structure models, experimentally determined material properties and finite element analysis are required, all of which contribute significantly to the accuracy of solder joint life prediction. In the past, many intrinsic structure models have been proposed for solders. For example, much work has been done to capture a wide range of temperature and strain rates in the intrinsic structure model[17]. These models eliminate the inconsistencies present in conventional models and rationalize the prediction framework by interlinking parameters. A new acceleration factor equation is also proposed to predict the reliability of solder joints considering the thermal cycling rate. Since it is not reasonable to include all influencing factors in a single physical framework, there is a huge gap between model results and experimental tests. In addition, some statistical methods, such as the Weibull distribution[19], have been applied to the prediction of solder joint life. Weibull distribution is the theoretical basis of reliability analysis and life testing, Weibull distribution predicts the percentage of failed solder joints during the aging phase, the percentage of failed solder joints during the active life phase, and when rapid aging will occur. However, Weibull distribution may significantly underestimate or overestimate reliability and mean life of solder joint to failure[2].

Reviewing the existing literature, the experimental methods and achievable models for solder joint reliability assessment are time-consuming and the predicted results are not accurate. Therefore, artificial intelligence can be a promising alternative solution for solder joint reliability assessment. In recent years, the use of machine learning (ML), such as neural networks (NN), has increased dramatically in various technical areas. Data analytics, speech or facial recognition, and autonomous driving are prime examples of modern machine learning use cases. ML can be used not only for consumer products but also for academic research and reliability analysis. Adequate training data sets create opportunities to develop predictive models for mission-critical system reliability assessment. And it has been demonstrated in several previous papers that failure modes predicted by artificial intelligence mechanisms are much more accurate than traditional statistical methods. However, in the literature, examples of FEA replacement or life prediction can be found, but these examples are rare.

In this paper, we propose a machine learning model to predict the reliability of solder joints under isothermal vibration loading with different geometries. Temperature, amplitude, the thickness of PCB and chip, solder joint volume(diameter), and solder joint count are used as input. And the equivalent elastic strain on the solder joint is used as output instead of the solder joint useful lifetime. For the solder, the previous paper[18] proved that SAC305(Sn96.5Ag3.0Cu0.5) is the most cost-effective. In addition, we use FEA simulation data and discuss how much data is needed for a given complexity of the model, what accuracy can be achieved for simulation results. Noise is also added to the training dataset so that the network has resistance to potential errors in experimental data.

2 Fundamental Theory Of ML Neural Networks

2.1 Artificial Neural Network

Artificial neural network (ANN) are computational systems inspired by the biological neural networks that make up the brain. ANN can enhance functionality such as adaptation or learning, fault tolerance, input-output mapping, non-linear functionality and lower-level functionality such as pre-processing of different types of data Inputs. It's based on collections of connecting units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like a synapse in a biological brain, can transmit signals to other neurons. The artificial neuron receives the signal, processes it, and can send signals to the neurons to which it is connected. The "signal" at the connection is a real number, and the output of each neuron is calculated as a nonlinear function of the sum of its inputs. These connections are called edges. Neurons and edges usually have weights that are adjusted as learning proceeds. The weights increase or decrease the signal strength at the connections.

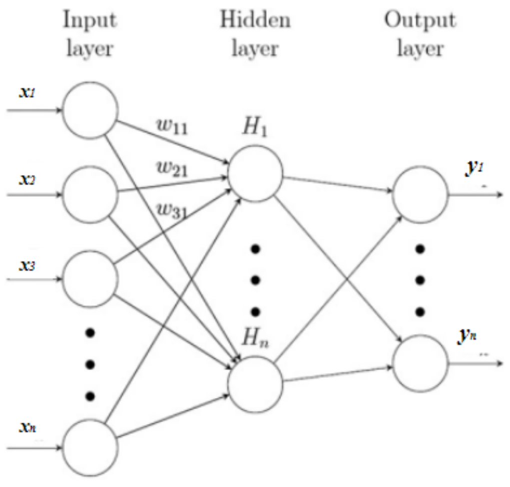


Figure 2-1: Schematic diagram of neural network

Fig2-1 shows a typical structure of an ANN model, where the left column is the input layer, the rightmost column is the output layer, and between the input and output layers are hidden layers. Often there may be more than one hidden layer. A neuron is a processing element (PE) whose output is computed by multiplying its inputs by a vector of weights, summing the results and applying an activation function to the sum. Each processing element has several inputs and one output. The relationship between the input x and the output y of a single processing element can be expressed as:

$$y_i = f(\sum_{j=1}^{m} w_{ji}x_j + b_i) \tag{2-1}$$

where w_{ji} are weights, b_i is a constant and normally referred to as bias. m is the number of inputs. $f()$ is called the activation function and is usually chosen from a list of S-shaped functions, such as sigmoid, hyperbolic tangent and Rectified Linear Units(RELU). The data flow in this type of network is from the input layer to the hidden layer and then to the output layer.

Fig.2-2 illustrates 2 common activation functions. In a neuron, the input features are weighted, summed, and then also act on a nonlinear activation function, allowing the neural network to approximate any nonlinear function at will, increasing the complexity of the network

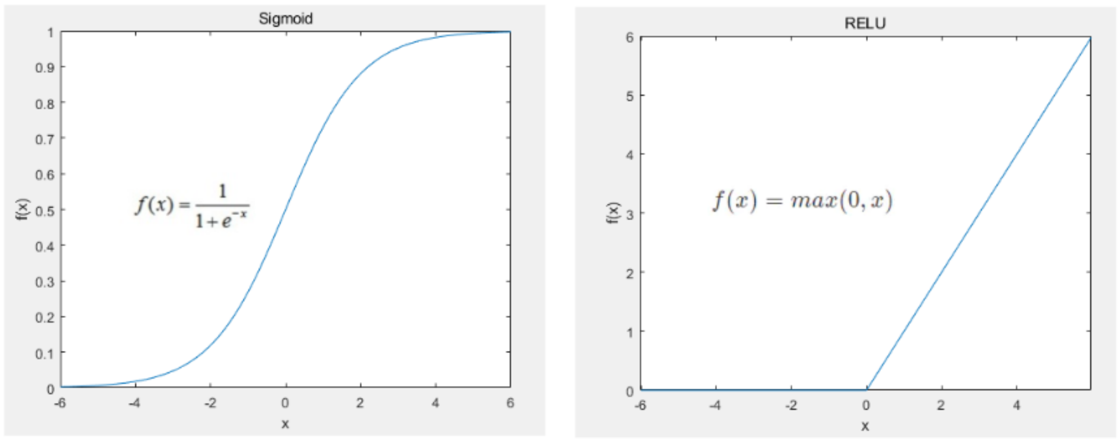


Figure 2-2: 2 common activation functions.

and thus avoiding that the output of each layer is a linear function of the input of the previous layer.

The performance of the identification models are carefully assessed through the lenses of the model precision, The performance of the regression models is evaluated using the mean square error (MSE), which is also called the Loss. Eq.2-2 shows the Loss function:

$$MSE = \sum_{i=1}^{m} \frac{1}{n}(\hat{y}_i - y_i)^2 \tag{2-2}$$

where $\hat{y}_i$ is predicted value and y_i is the actual value of the output. Loss will be used as input for subsequent backpropagation to update the values of weights and biases.

So the model of a ML neural network is a functional relationship that shows the mapping of input data to output data, and the process of model training is actually learning the corresponding values of weights and biases. In contrast to the weights and biases that can be learned automatically, another kind of parameters are called hyperparameters, such as the learning rate, the number of hidden layers, the number of neurons per hidden layer and the number of epochs. These hyperparameters are pre-set, which determine the structure of the network and cannot be learned automatically, while the tuning of the model actually refers to adjusting the hyperparameter.

2.2 Standard Backpropagation Algorithm

As mentioned in the previous chapter, the data is forward propagated through the neural network to get the Loss, but usually this Loss will be large. What we have to do is to find a way to keep shrinking the loss until it reaches the theoretical minimum. This process is called 'train' in machine learning. The most effective method currently recognized is backpropagation. Backpropagation (BP), short for "error backpropagation", is a common method used in conjunction with optimization methods such as gradient descent and least squares to train artificial neural networks. This method calculates the gradient of the loss function for all the recalculations in the network. This gradient is fed back to the optimization method and used to update the weights to minimize the loss function. The main algorithm for performing the gradient descent method on a neural network. The algorithm first calculates (and caches) the output values of each node in a forward propagation manner, and then calculates the partial derivatives of the loss function values with respect to each parameter in a backward propagation traversal graph.

The learning process of the BP algorithm consists of a forward propagation process and a backward propagation process. In the forward propagation process, the input information is processed through the input layer via the hidden layer, layer by layer and passed to the output layer. If the desired output value is not obtained in the output layer, the sum of the squares of the output and the desired error, i.e., Loss, is taken as the target function and transferred to the backward propagation, where the partial derivatives of the target function on the weights of each neuron are derived layer by layer, constituting the gradient of the target function on the vector of weights, as the basis for modifying the weights, and the learning of the network is completed in the process of modifying the weights. The network learning is completed in the process of weight modification. The network learning ends when the error reaches the desired value or after a certain time and epoch.

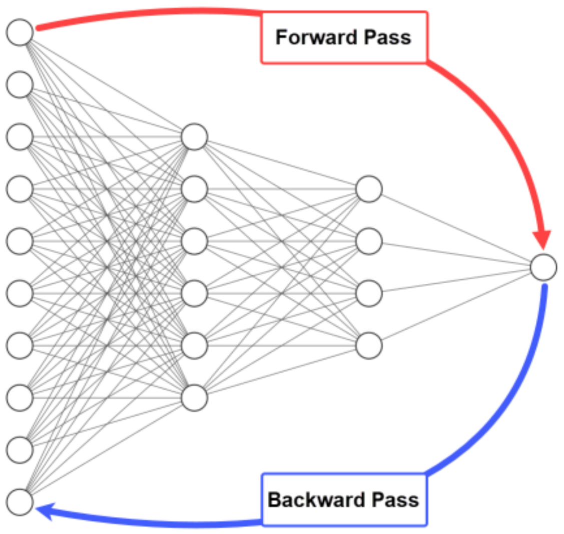


Figure 2-3: Forward- and backwardpropagation

Fig.2-3 shows forward- and backwardpropagation of the NN. From the figure, we know that the backpropagation is to transfer the signal from the output layer to the input layer, and the backpropagation input at this time is the Loss calculated after one forward propagation, which is written here as δ. On each neuron is the calculated error response ζ . ζ can be expressed as:

$$\zeta_i = f(\sum_{j=1}^{m} w_{ji}\delta_i + b_i) \tag{2-3}$$

In order to minimize the error response of each neuron, the partial derivatives of the error responses to the weights and biases need to be known. According to the chain rule, the formula for the partial derivative can be expressed as :

$$\begin{aligned} \frac{\partial \zeta}{\partial w} &= \frac{\partial \zeta}{\partial \delta}\frac{\partial \delta}{\partial y}\frac{\partial y}{\partial w} \\ \frac{\partial \zeta}{\partial b} &= \frac{\partial \zeta}{\partial \delta}\frac{\partial \delta}{\partial y}\frac{\partial y}{\partial b} \end{aligned} \tag{2-4}$$

Next, the weights and biases are updated using different optimization algorithms. Here is an example of the gradient descent:

$$\begin{aligned} w_{new} &= w_{old} - \alpha\frac{\partial \zeta}{\partial w_{old}} \\ b_{new} &= b_{old} - \alpha\frac{\partial \zeta}{\partial b_{old}} \end{aligned} \tag{2-5}$$

where α is learning rate and usually takes the value of 0.01.

2.3 Training Process

The training process is actually updating the values of weights and biases in each epoch to minimize the Loss. Through continuous trial and error until optimal. Fig2-4 illustrates a complete 'train' process:
1) Forward propagation of data to calculate Loss.
2) The loss is back-propagated as an input and the error response of each neuron is calculated, along with its partial derivatives with respect to the weights and biases.
3) Update the weights and biases using the optimization algorithm.
4) Repeat 1), 2), 3) steps with the updated weights and biases until Loss reaches the desired value.

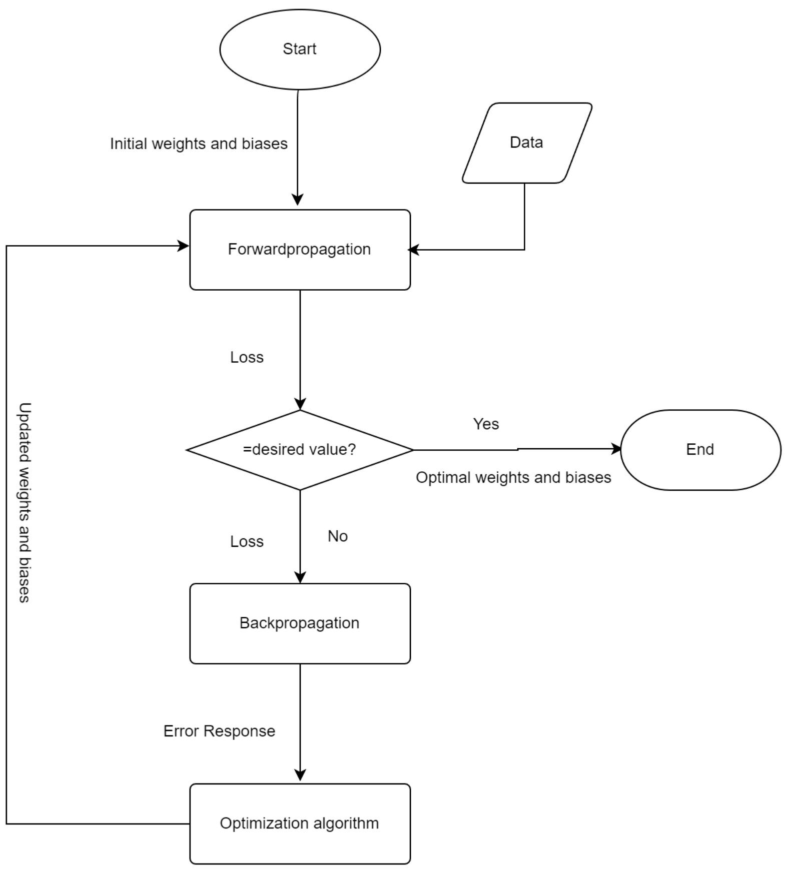


Figure 2-4: Workflow of the 'train' process

3 Literature Research

This section aims to find use cases that use machine learning to predict the reliability of solder joints. However, no publications were found that used machine learning to predict the reliability of solder joints under isothermal vibration. Therefore, an attempt was made to investigate the selection of features and the structure of the neural network under temperature cycling conditions, and trying this out for the isothermal vibration load case.

Fig.3-1 shows the relation between the measured useful lifetime(UL) and 3 T.C. specifications, i.e. dwelling temperature, dwelling time and ramp rate, which are significantly affecting the reliability of a solder joint under a thermal cycling process. As observed in the three equations in the figure, the rate of heat-up with a higher slop significantly affects the reliability of solder joints in electronic devices, while the dwell time and temperature parameters have a more moderate effect on UL. In addition, the trend of the distribution of points shows that the points in the ramping-UL plot shrink in a smaller ellipse compared to the other two figures. This is because the dwelling temperature and time in the thermal cycling profile are strongly dependent on each other and on the material properties of the components in the electronic device. Typically, thermal cycling processes lead to creep and fatigue in solder joints. In this case, the thermal dwelling portion of the thermal cycle, including the dwelling time and its corresponding temperature, is indicative of a creep failure, while the ramp portion leads primarily to fatigue failure[20, 1]. It is worth noting, however, that creep and fatigue in thermal cycling are intertwined and difficult to distinguish. The more pronounced effect of ramp rate on UL does not necessarily indicate a dominant role of fatigue in failure occurrence.

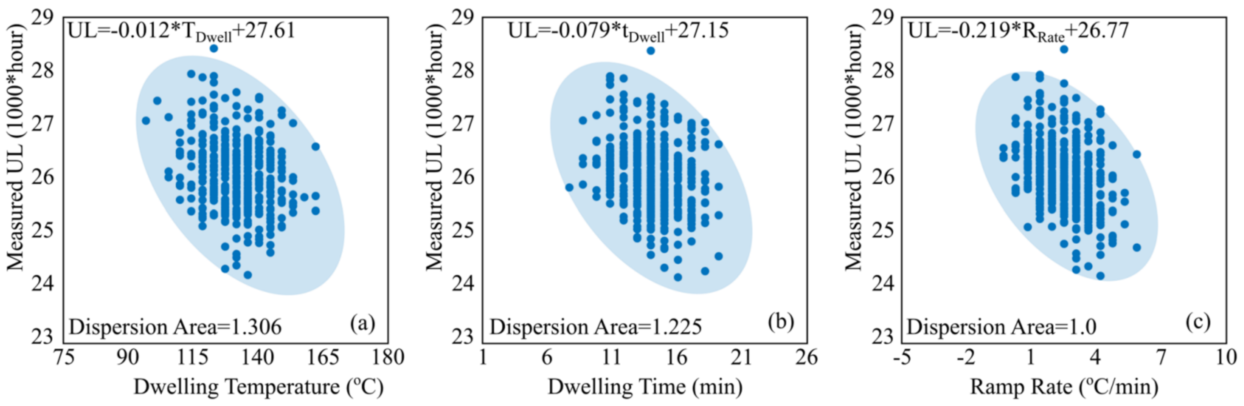


Figure 3-1: Relation between the measured UL and 3 T.C. specifications .. (a) Useful lifetime versus dwelling temperature, (b) useful lifetime versus dwelling time and (c) useful lifetime versus ramp rate.[21]

The other important feature candidate is the joint zone material properties[23, 4, 24, 12], like Young's Module, Density, Poisson ratio, etc. For instance, Heinrich[12] presented that the properties of the different materials can significantly impact the deformation of the joint zone. And the degree of deformation affects the useful lifetime of electronic devices. In the paper, he proposed a factor called the Shear correction factor, which depends on the properties of the solder joint, chip, and PCB, to describe the deformation of the solder joint. That means, under the same thermal cycling conditions, the differences of the useful lifetime between the different material of solder joint is obviously large.

Moreover, the geometry size of the solder joint is also an important factor in deformation[9, 16, 5]. For instance, Liang[16] proved through experiments that the geometry size has a significant impact on the reliability. As is shown in Fig3-2, when a solder joint's height is reduced from about 0.54 mm to 0.32 mm, the fatigue life can drop by more than 90%.

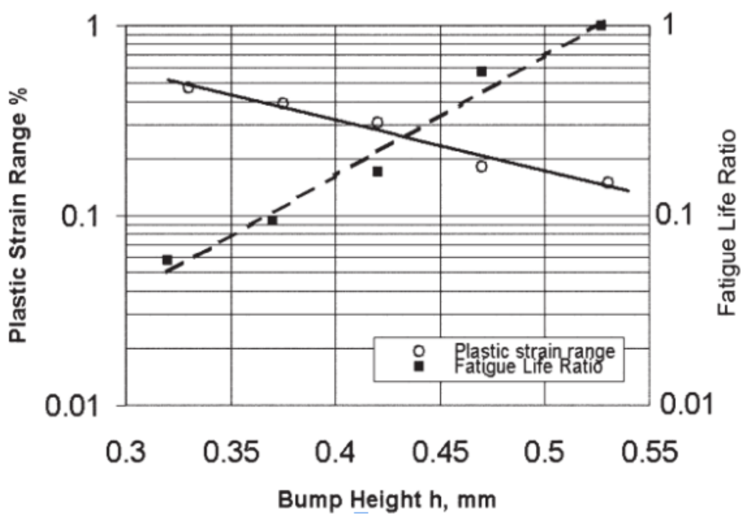


Figure 3-2: Solder bump height versus plastics strain and fatigue lives.[16]

For the structure of the NN, the conventional artificial neuron network(ANN) and some novel NN are used in the prediction model. Sung Yi and Robert Jones[26] present that the predicted failure mode via artificial intelligence mechanisms is much more accurate than the conventional statistical methods(the Weibull method). The result shows in Tab3.1. In this paper, they consider that different surface finishes in copper pads significantly affect reliability. Therefore, for the NN structure, the input is the 3 different surface finish treatments, namely electroless nickel immersion gold(ENIG), immersion silver(IAG), and organic solderability preservative(OSP). They use One-hot encoding to break up these 3 treatments into three binary attributes. And the output is the number of cycles to failure. Also, there are 3 hidden layers each with 50, 100, and 50 neurons. And it's worth mentioning that they improve the performance of the NN and accelerate the computation speed by using Xavier initialization to initialize the weights and using the Adam[15] algorithm to compute the weight and bias corrections in backpropagation.

Surface treatment	Average errors(%)(ML)	Average errors(%)(Weibull)[10]
IAG	5.2	22.0
ENIG	5.1	11.7
OSP	5.0	21.5

Table 3.1: Average errors for predicted failures[26]

Samavatian[21] presented a new research direction suggesting to use a correlation-driven neural network (CDNN)(Figure3-3) proposed for predicting the useful lifetime of the solder joint. In the paper, as is shown in Figure3-4, 3 distinct categories are considered as the input of the CDNN, which significantly impact the solder joint reliability, namely the thermal cycling specifications, Joint zone specifications, and Geometry size. In the model, there are two unique paths and follow by a deep neural network. The first path with the method cross-correlation is aimed to capture the mutual impacts of the main features during the training process. The second path directly transfers the feature to the deep neural network. Accordingly, the

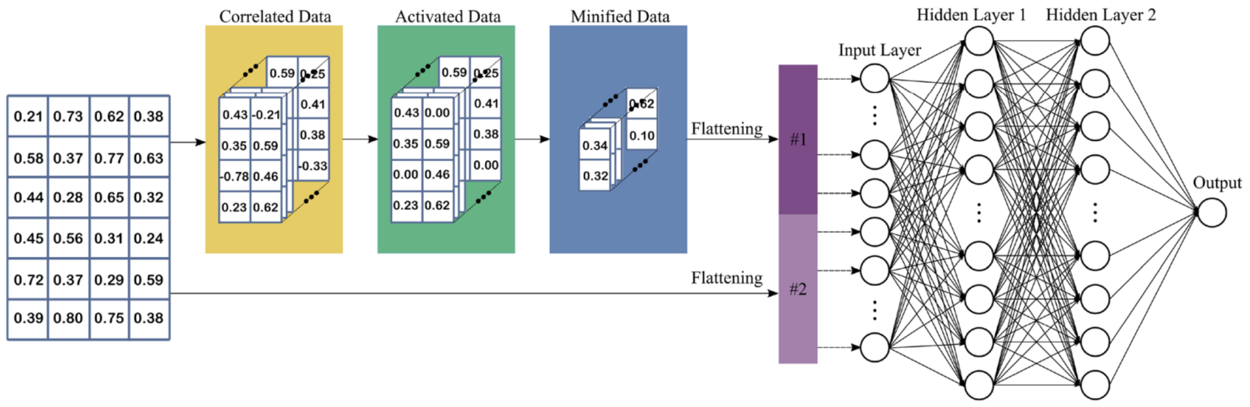
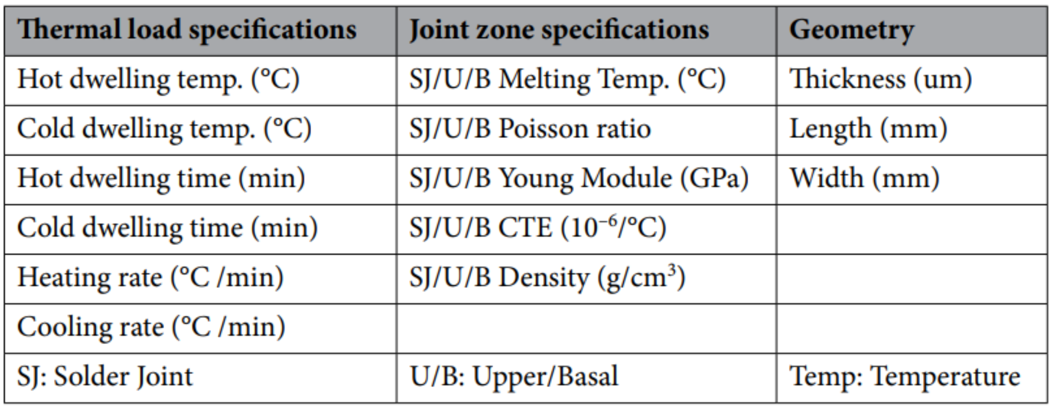


Figure 3-3: CDNN model.[21]

individual and mutual interactions of these feature candidates have to be captured in ML mechanism to accurately predict the useful lifetime of the solder joint. The result shows in



Thermal load specifications	Joint zone specifications	Geometry
Hot dwelling temp. (°C)	SJ/U/B Melting Temp. (°C)	Thickness (um)
Cold dwelling temp. (°C)	SJ/U/B Poisson ratio	Length (mm)
Hot dwelling time (min)	SJ/U/B Young Module (GPa)	Width (mm)
Cold dwelling time (min)	SJ/U/B CTE (10^{-6}/°C)	
Heating rate (°C /min)	SJ/U/B Density (g/cm^3)	
Cooling rate (°C /min)		
SJ: Solder Joint	U/B: Upper/Basal	Temp: Temperature

Figure 3-4: The physical properties, thermal load specifcations and geometry features.[21]

Fig3-5. It's obvious that the CDNN model has a better performance than the CANN model.

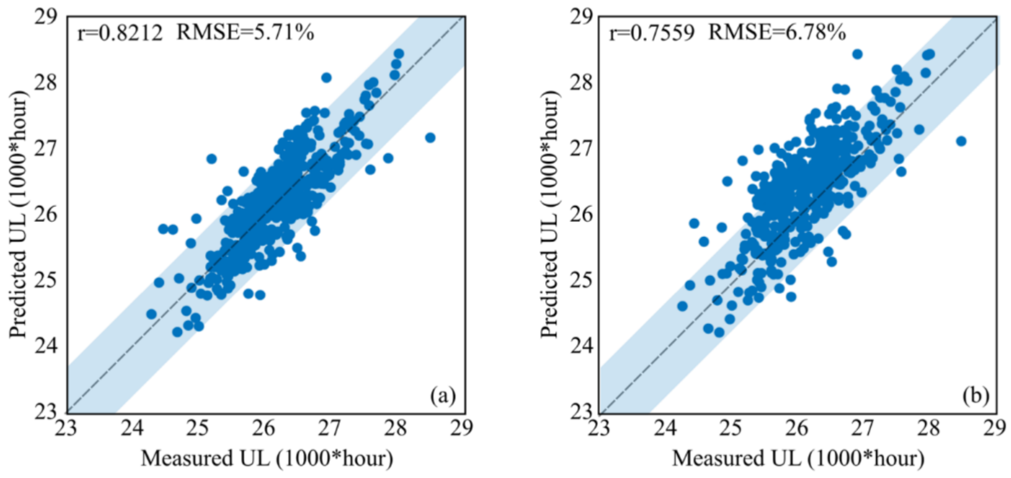


Figure 3-5: Neural network predicted values versus the measured values. (a) Proposed correlation-driven neural network (CDNN), (b) Conventional artificial neural network (CANN).[21]

Then, Vahid[22] proposed an iterative ML framework to improve the performance of the CDNN model. The input data in this paper is the same as the CDNN. This improvement in the prediction accuracy is achieved by injecting the different thermal cycling specifications into the initial data set. Because the CDNN model cannot capture low-/high-intensity thermal loading conditions regarding the sparse input data.

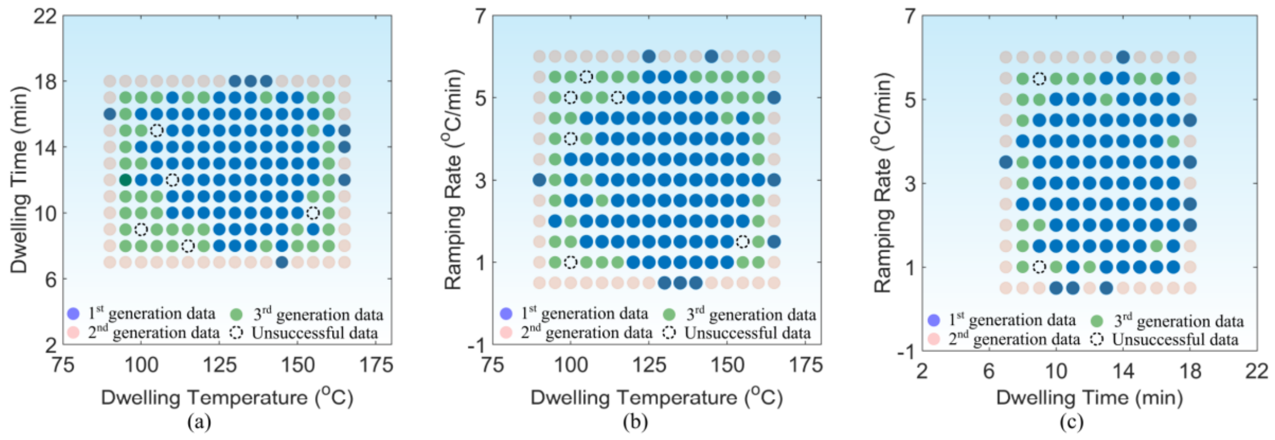


Figure 3-6: The dispersion of three major thermal loading specifications.[22]

Fig3-6 shows the dispersion of three major thermal loading specifications. The blue-filled circles are the initial data, which concentrate in the middle area. And there exists a lot of missing data points in the data region. The red- and green-filled data are the injected data, which are injected into the position of the missing data points. The specific iterative process is :
1) Fill the boundary data(red-filled data) and use FEM to measure the useful lifetime of these data.
2) Train the CDNN model with the blue- and red-filled data. And predict the useful lifetime of the green-filled data with the trained model. Then, compare the predicted useful lifetime of the green-filled data with the useful lifetime of the vicinity of the input(red- and blue-filled

data).
3) Check if the error of the comparison is less than 5%, mark it as successful data if it is, and mark it as unsuccessful data if it is more than. Then, inject the successful data into the input. Then repeat steps 2 and 3.
The result shows in Fig3-7, Compared with the CDNN model the iterative CDNN model has a higher correlation rate(r=0.9784) and lower root mean square error(RMSE)(2.12%).

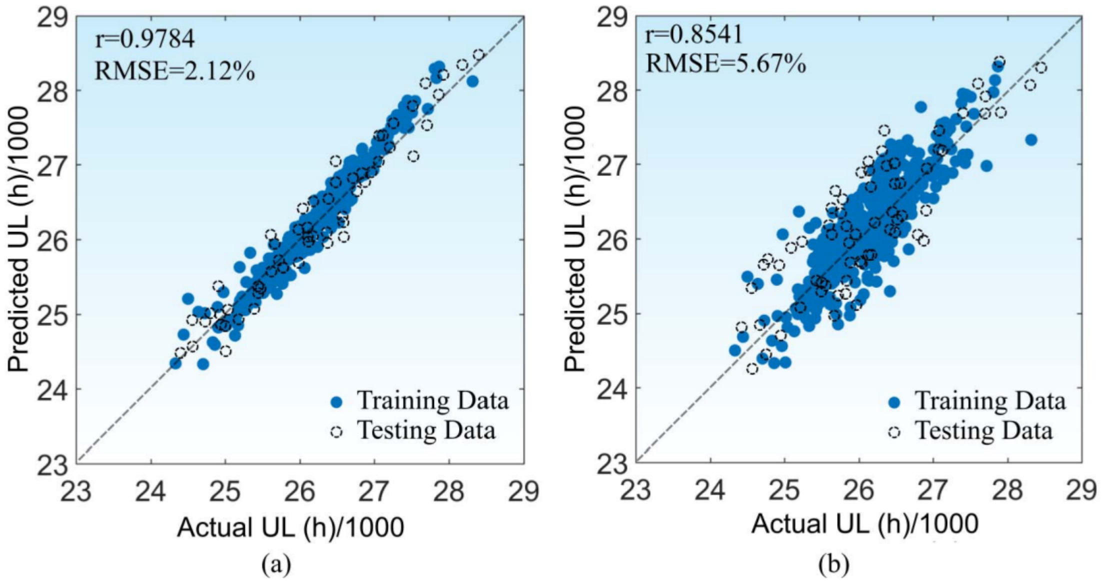


Figure 3-7: Neural network predicted values versus the actual values. (a) Iterative CDNN, (b) CDNN.[22]

Recently, Chen[7] presented that there exists an optimum solder joint volume from a reliability perspective which can be considered during the device designing and manufacturing, and the type of the solder joint(Barrel type and hourglass type) can also significantly impact the reliability. So, for the feature candidates, they consider the thermal cycling specifications, solder joint volume, and the solder joint type. A conventional ANN with 3 hidden layers is employed, each layer has 50 neurons. The result shows in Fig3-8. It's evident that, there exist an optimum solder joint volume in reliability perspective, and hourglass solder joint has a longer lifetime in comparison with barrel type solder joint under the certain conditions.

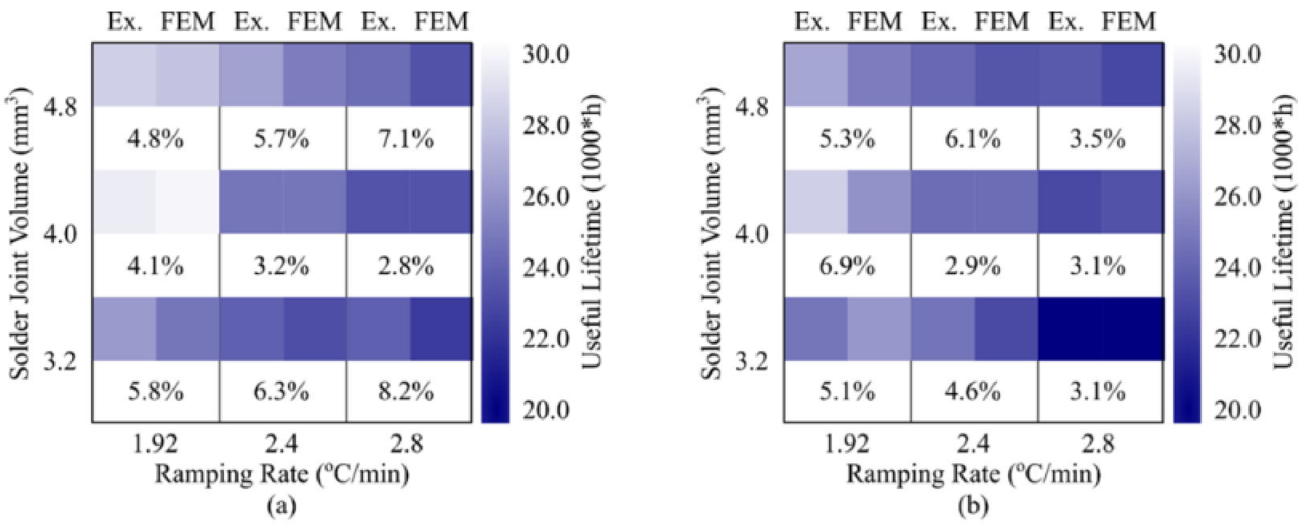


Figure 3-8: Simulation and experimental results under the same conditions. (a) estimated hourglass type solder joint useful lifetime, (b) estimated barrel type solder joint useful lifetime.[7]

And in Chen's other paper[8], he proposed a novel approach to predict the reliability of the solder joint. Compared with the input of the CDNN model, only the solder joint properties were considered. However, in terms of innovation, radial basis function(RBF) NN was chosen. The structure of the RBF NN is shown in Fig3-9(a). In contrast to CANN, RBF NN has only a single hidden layer with a nonlinear radial basis function, i.e. Gaussian function, and this is where the power of the RBNN network originates. The result shows in Fig3-10. In comparison with Conventional Neural Networks(CNN) with an error of 7.35%, the proposed radial-based neural network had a 3.1% error in its prediction.

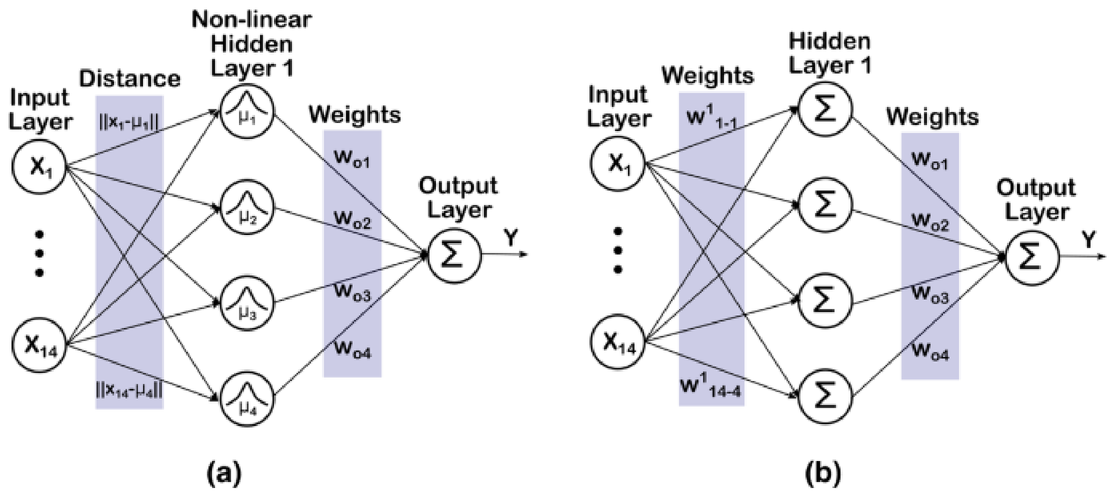


Figure 3-9: Studied neural network architectures for solder joint lifetime estimation: (a) RBNN structure, (b) CANN structure.[8]

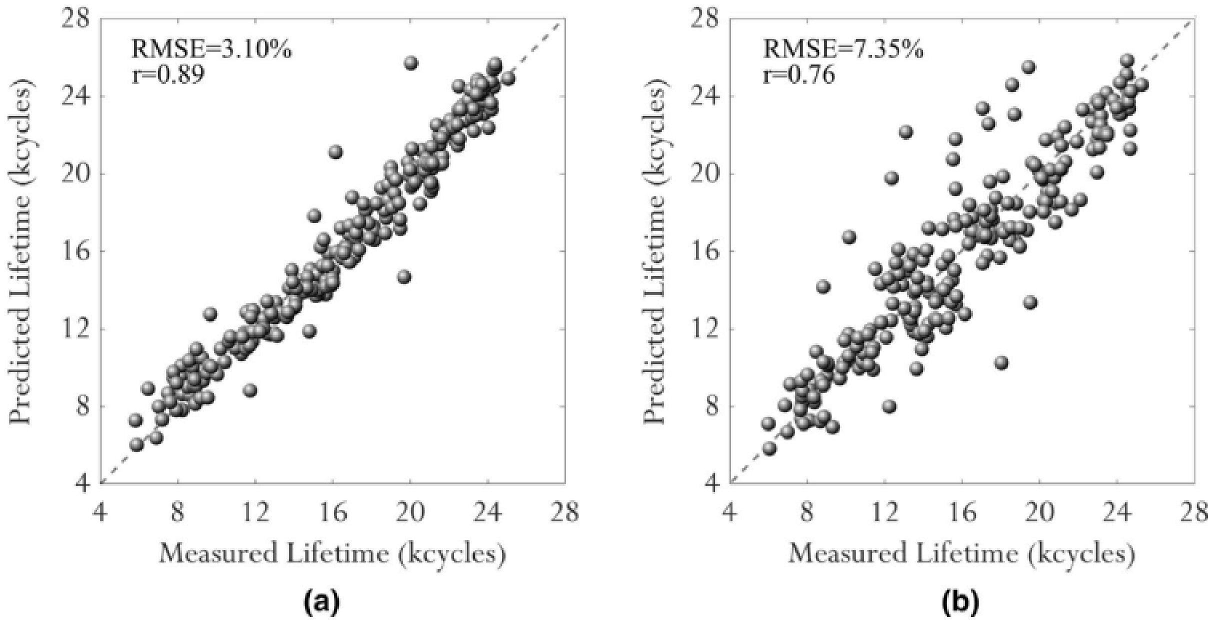


Figure 3-10: Performance of predictive models for solder joint useful lifetime estimation: (a) proposed RBNN predictive model, (b) CNN predictive model.[8]

4 FEA Model

As shown in Fig.4-11, our model is specifically designed for isothermal vibration. In the vibration simulation, the entire spine in the middle is fixed, and the very top and bottom (wider parts) are used for the electrical connection during experiments. The accelerometer are placed on the left and right ends. The position close to the spine has a larger strain and smaller amplitude, and the position at the accelerometer has a larger amplitude and smaller strain. At the same time, the higher the temperature, the more flexible the substance is and the greater the strain, and the lower the temperature, the more stubborn the substance is and the smaller the strain. Fig.4-11 shows the arrangement and location of the solder joints and flip chips, which have up to 25 solder joints and are fully populated. As is shown in figure, the chip, solder joint and PCB are near the spine, which happens to be where the strain is greatest.

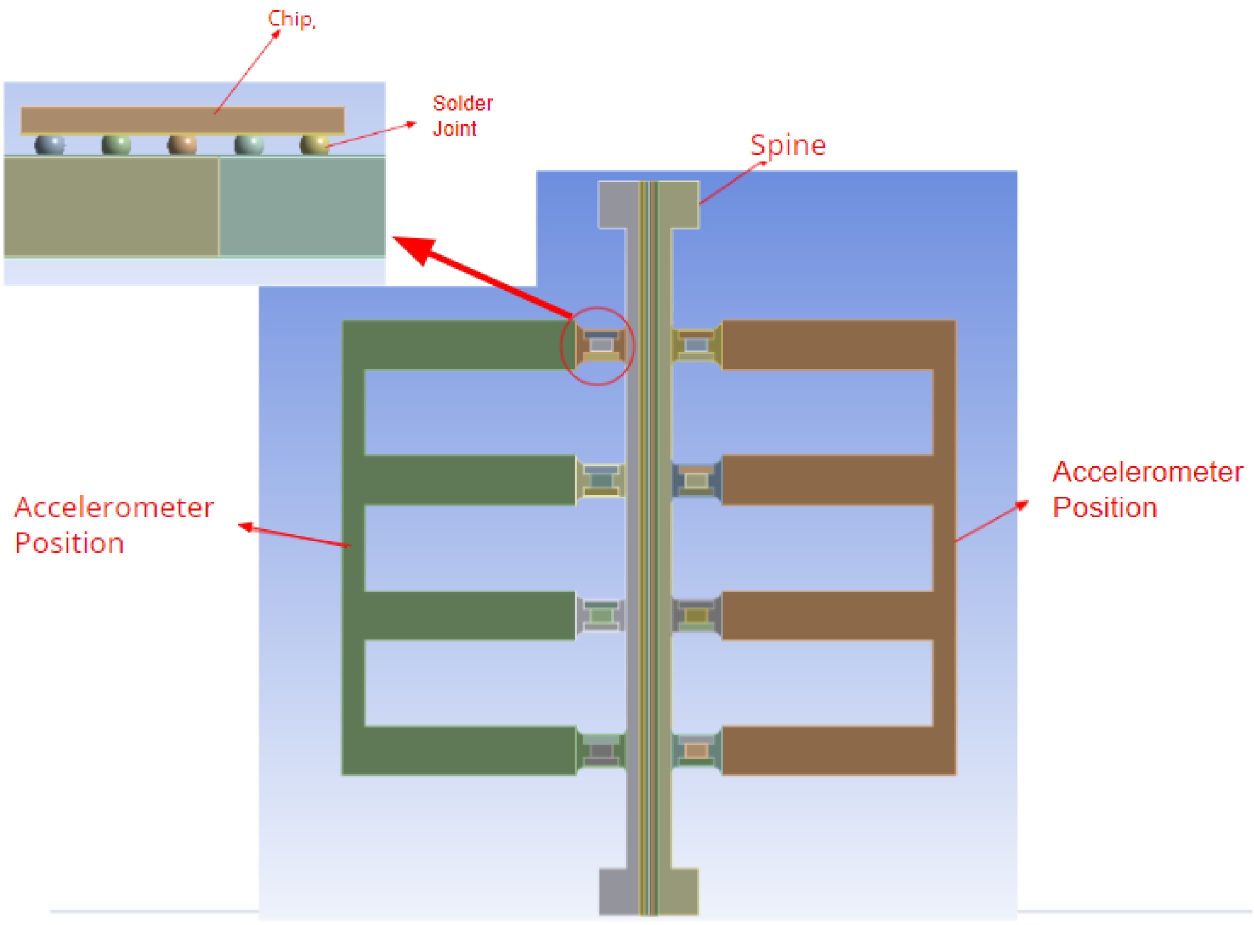


Figure 4-11: Specimen in simulation and Enlargement at the red circle.

In the vibration experiment simulation, the frequency of vibration is fixed at 125 Hz and different results are calculated by varying the temperature, amplitude and geometry. We ultimately want to obtain the maximum elastic strain observed with each set of parameters. However, it takes a lot of time to complete a complete simulation, and subsequent machine learning has the need for hundreds or thousands of data. Previous studies have shown that using a half or even quarter model is a adequate solution to cut simulation time. Therefore, as shown in Fig.4-12, we simulate only a quarter of the model and calculate the equivalent elastic strain on the first solder joint only.

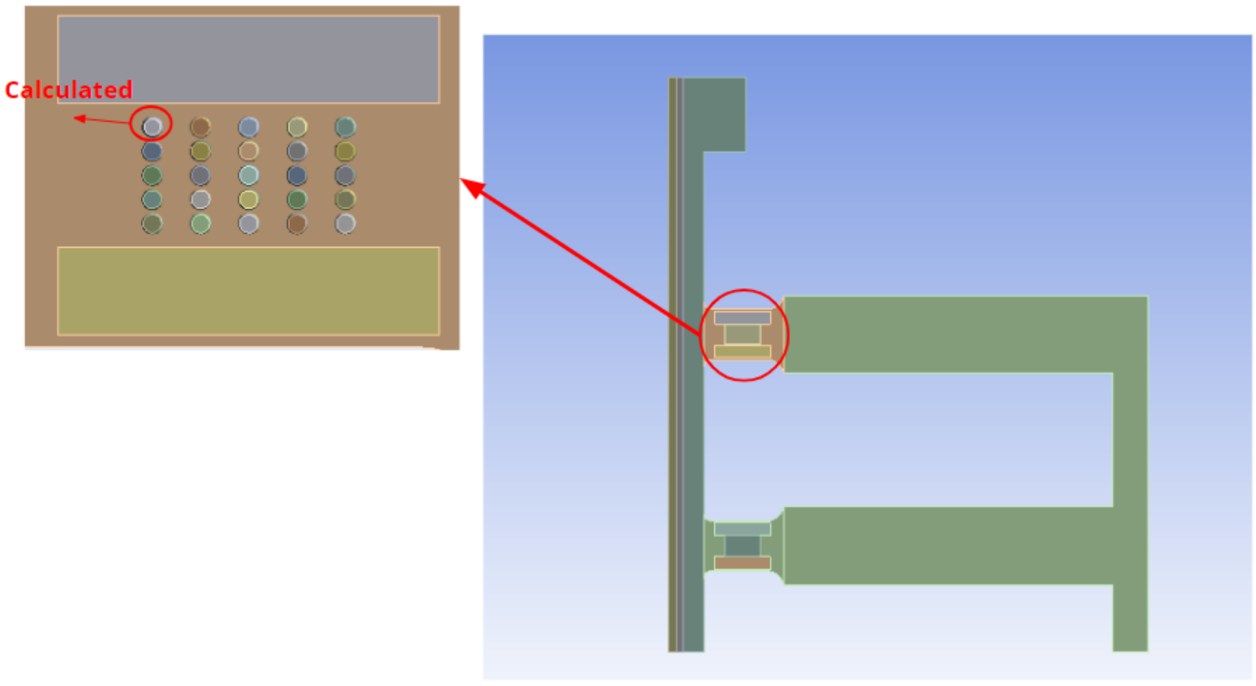


Figure 4-12: Quarter of the specimen and One solder joints for calculation.

5 Implementation

It is worth stating that all the code in this paper is developed and run based on the Matlab Deep Learning Toolbox environment. The code for all implementation sections including data processing, neural network building, adding noise, and model training are documented in detail in the appendix.

5.1 Feature Candidates

Candidate features (X) are considered as inputs to the machine learning model (f) to accurately predict the desired output (Y), i.e., Y=f(X). Therefore, a sufficient set of X must be defined for a given target output Y to ensure that a well-performing machine learning model is generated.

In conjunction with the literature reviewed in the 'Literature Research' section, we identified six features that have a significant impact on the simulation results, i.e., temperature, amplitude, PCB thickness, chip thickness, solder volume, solder count. As for other larger influencing factors, such as chemical composition and physical properties of PCB, chip, and solder joint, we strictly fixed them, so they are not considered. Tab.5.1 shows the inputs and outputs in the machine learning model.

Input	Output
temperature	
amplitude	
PCB thickness	
chip thickness	equivalent elastic strain
solder volume	
solder count	

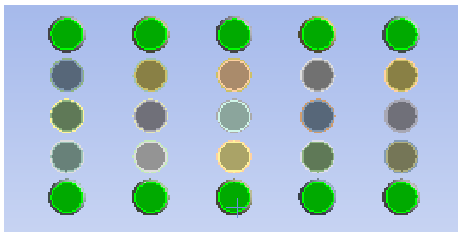
Table 5.1: Input and output of the ML model.

It is worth mentioning that the solder count, one of the input featuress, is a very unique aspect compared to the solder volume or thickness. It shows the arrangement of each solder joint under the chip. As is shown in Fig.5-13, the reduction of the number of solder count is not just the removal of the solder joints, but also the change of the pattern of the solder joint arrangement.

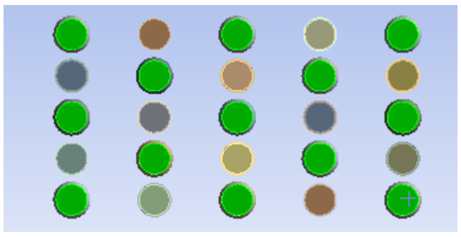
However, it may not be appropriate to include the data directly in a neural network or any other machine learning algorithm. This is because the range of variables varies greatly in the raw data. For some machine learning algorithms, the objective function will not work properly if it has not been normalized. In response, data preprocessing, i.e., feature scaling, is chosen to normalize the feature candidates. The candidate features are rescaled to a predefined range of [a, b] using the min-max scaling formula. This formula can be expressed as:

$$x_{new} = a + \frac{(b-a)(x - min(x))}{max(x) - min(x)} \tag{5-1}$$

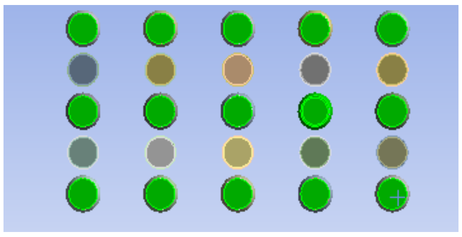
where x and x_{new} are the original feature and scaled feature candidates, respectively. $min(x)$ and $max(x)$ are the minimum and maximum values of the feature candidates on the training data set, respectively. An appropriate scaling range has been reported to achieve higher accuracy and faster Loss convergence. Therefore, we consider $a = 0.2$ and $b = 0.8$[25].



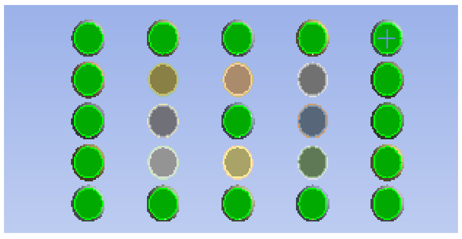
(a)



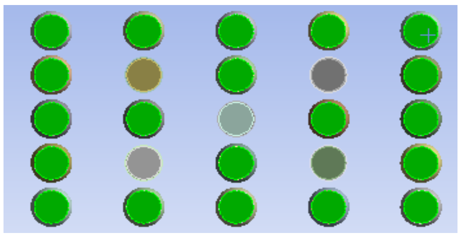
(b)



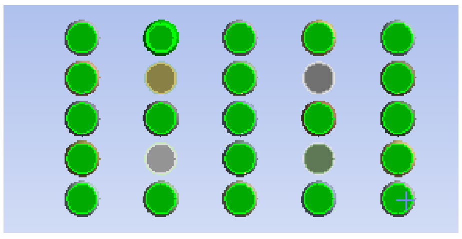
(c)



(d)



(e)



(f)

Figure 5-13: The pattern of the solder joint arrangement, where the green part is the solder joint. (a)10 solder joint, (b)13 solder joint, (c)15 solder joint, (d)17 solder joint, (e)20 solder joint, (f)21 solder joint.

5.2 Data Collection

Collecting sufficiently large input data plays an important role in the training process. In principle, the larger the amount of data, the more accurate the prediction results. However, too large a dataset may contain too much homogeneous data, which may cause overtraining, and producing large amounts of data takes huge amounts of time. So subsequently we will look for a suitable number of data sets, where the relative error of the measured and predicted value of the dataset will converge. For this purpose we have provided 500 sets of data for simulation, all collected data have been extracted from the finite element model (FEM) simulation results and the units of the parameters have been standardized.

In a previous literature search, we learned about some of the parameters that affect the aging of solder joints. So here we used six different features to characterize the input factors used to train the prediction model. Each of these features contains several parameters related to solder joint degradation. Several key features may have a serious impact on the lifetime of a solder joint. These features include temperature, amplitude, thickness of the PCB, thickness of the chip, volume of the solder joint, and count of the solder joint. These six features do not correlate with each other.

Tab.5.2 records these six different features and their value range. Theoretically, the combination of these values has about 44,800 results. Since the simulation is isothermal vibration, there are certain temperatures and amplitudes in each simulation, each value on them is sampled randomly and uniformly when generating the dataset. However, a new FE model had to be created for each change in geometry, which was very time consuming and well beyond the scope of the student project, we cannot combine the geometries so randomly that most of the values are concentrated in one value, e.g. 1.5mm for PCB thickness, 0.4mm for chip thickness, 300μm for solder joint diameter(here a diameter is used to represent a specific volume), and 25 for the count of solder joints. In other words, 1.5mm, 0.4mm, 300μm, and 25 are the standard values for geometry, and each time we change the value we can only change one of the values, while the other three remain the same. The results are displayed in Fig5-14, in the distribution of geometric values, most of them are concentrated in one value, and this distribution is similar to the normal distribution. The temperature, amplitude values and fixed geometric values based on the FE model are then simulated with a random combination , and the simulation results obtained are equivalent elastic strains. In the end, we simulated a total of 500 sets of data. Moreover, we also investigated the individual effects of each feature on the simulation results.

temperature(°C)	Amplitude(mm)	PCB thickness(mm)	chip thickness(mm)	solder volume (μm)	solder count
-40	0.05	1	0.3	225	10
-25	0.25	1.5	0.4	250	13
0	0.5	1.75	0.5	275	15
25	0.75	2	0.6	300	17
50	1		0.7	325	20
75	1.25				21
100	1.75				25
125	2				

Table 5.2: Input data and its value range.

Since the simulation result, i.e., the equivalent elastic strain, is too small, resulting in the MSE function as Loss is also too small to be used as the final evaluation metric of the model,

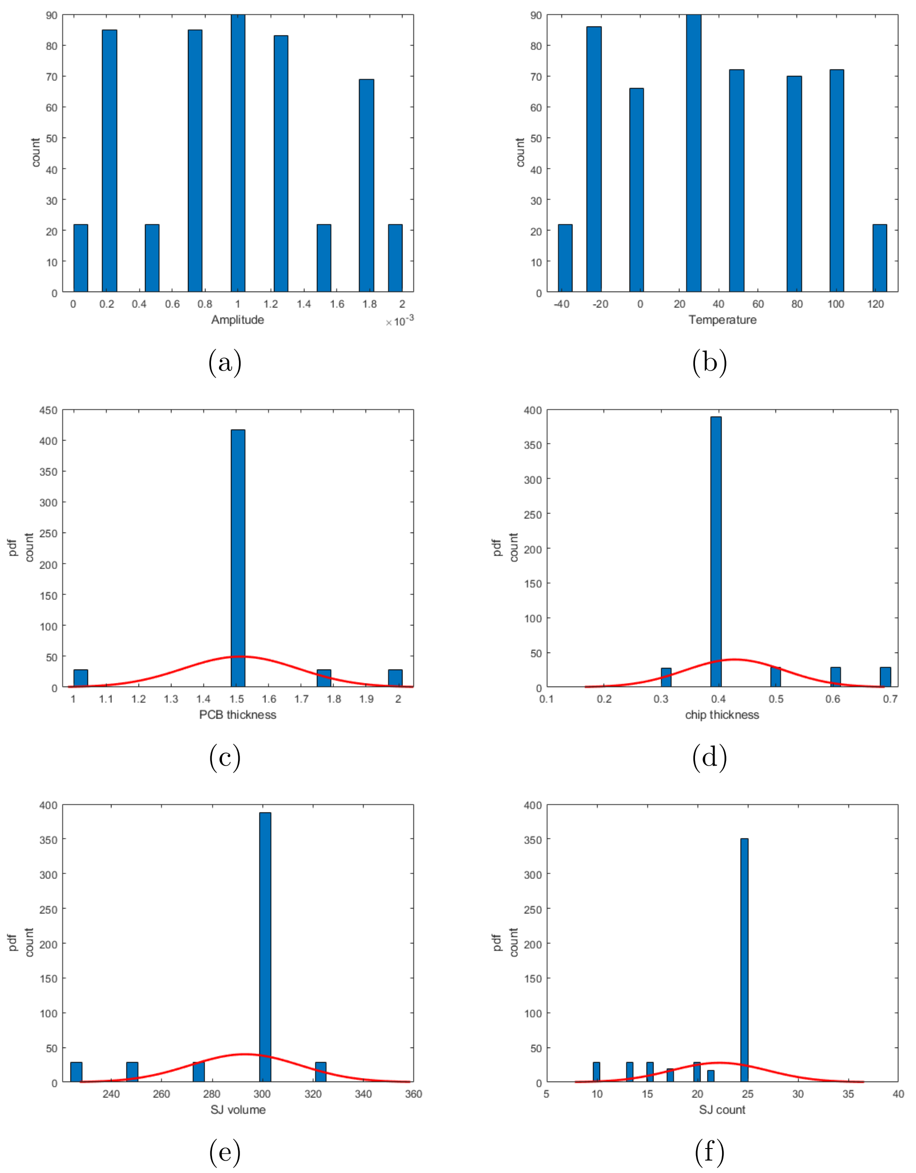


Figure 5-14: The distribution of (a)Amplitude, (b)Temperature, (c)PCB thickness, (d)chip thickness. (e)SJV, (f)SJC, the histogram indicates the number of each value, and the red curve is the normal distribution curve.

so here we use the relative error, which can be expressed as:

$$e = |\frac{y_{pre} - y_{mea}}{y_{mea}}| \times 100\% \tag{5-2}$$

where y_{pre} is the output value obtained from the network after training and y_{mea} is the measured value.

5.3 Setting Up The Neural Network

To build a neural network it is first necessary to determine its hyperparameters, i.e. the structure of the neural network (number of hidden layers, number of neurons on each hidden layer). The effect of the number of hidden layers on the neural network can be summarized in the following table:

Number of Hidden Layers	Result
none	Only capable of representing linear separable functions or decisions.
1	Can approximate any function that contains continuous mapping from one finite space to another.
2	Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.

Table 5.3: Determining the Number of Hidden Layers.

The deeper the layers, the better the theoretical fit and the better the results, but in practice, deeper layers may cause overfitting problems and also increase the training difficulty, making it difficult to converge the model. So here we choose two hidden layers.

Determining the number of hidden layers is only a part of the problem; one must also determine how many neurons are in each hidden layer. Using too few neurons in the hidden layer will lead to underfitting. Conversely, using too many neurons can also lead to several problems. First, too many neurons in the hidden layer can lead to overfitting. When a neural network has too many nodes (too much information processing power), the limited amount of information contained in the training set is not enough to train all the neurons in the hidden layer, thus leading to overfitting. Even if the training data contains enough information, too many neurons in the hidden layer will increase the training time, thus making it difficult to achieve the desired results. Obviously, it is crucial to choose an appropriate number of neurons in the hidden layer. There is a lot of experience on how to determine the number of neurons. In general, the number of hidden neurons should be between the size of the input layer and the size of the output layer[13]. In our dataset, the size of the input layer is 6 and the size of the output layer is 1, so the number of neurons in the hidden layer is approximately between 2 and 5. To determine the exact number of neurons, one can fix the number of neurons on the 2.hidden layers as 2, change the number of neurons on the 1.hidden layer. Then, the 500 previously collected data are trained and the relative error between the predicted and measured values is observed each time the number of neurons is changed, and the number of neurons corresponding to the smallest relative error is taken, as shown in Fig.5-15, which is 5. In the same way, the number of neurons in the 1. hidden layer is fixed to 5 and the number of neurons in the 2. hidden layer is tested. As shown in Fig.5-16, the number of neurons on the 2. hidden layer is determined to be 5. So, the optimal number of neurons of hidden layers is finally determined as [5 5]. In summary, the neural network structure is shown in Fig.5-17.

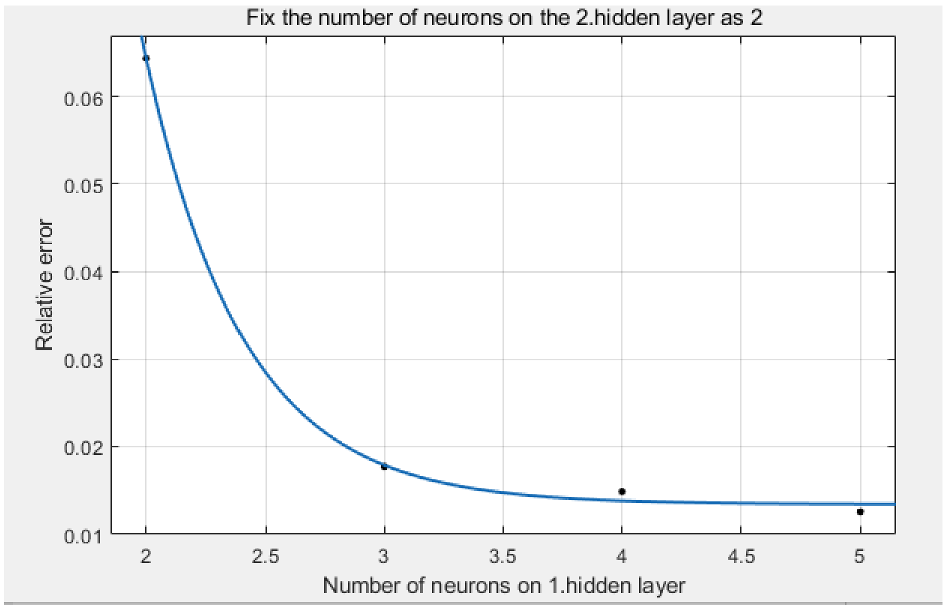


Figure 5-15: Number of neurons on 1.hidden layer vs. relative error with the number of neurons on the 2.hidden layer are fixed as 2.

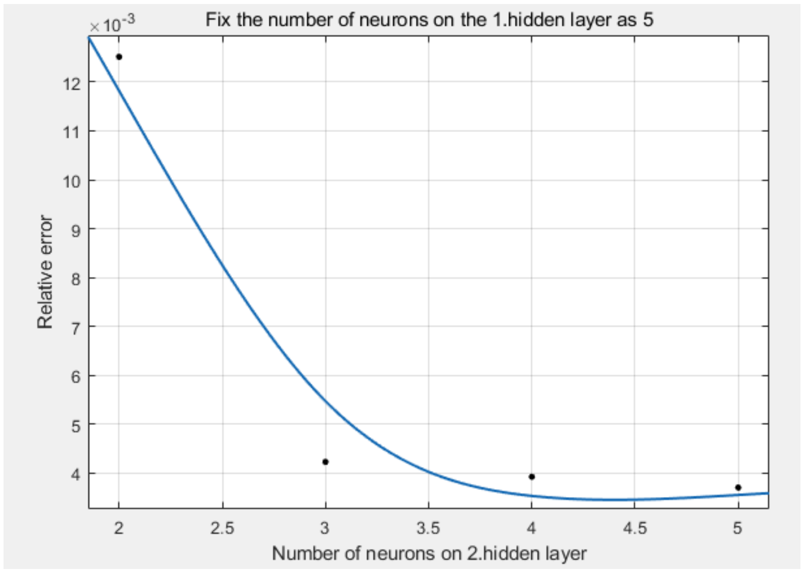


Figure 5-16: Number of neurons on 2.hidden layer vs. relative error with the number of neurons on the 1.hidden layer are fixed as 5.

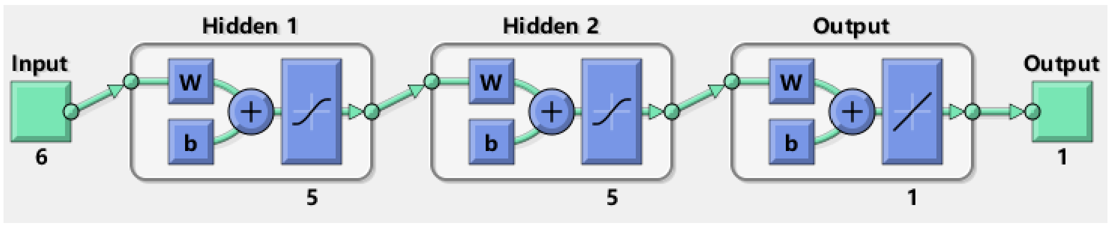


Figure 5-17: The structure of NN.

5.4 Determining The Size Of The Data Set

To determine whether 500 sets of data converge the relative error between predicted and measured values, 100, 200, 300, 400, and 500 sets of data are trained with the neural network. Fig.5-18 depicts the effect of different dataset sizes on the relative error between the measured and predicted value. It is clear that the relative error continues to decrease as the size of the dataset increases. At dataset size of 400 and 500, the relative errors are 0.4296 and 0.4186, respectively, and the variation between them is less than 1%, so it can be concluded that the curve converges at 400. Therefore, the 500 sets of data fully comply with the requirements.

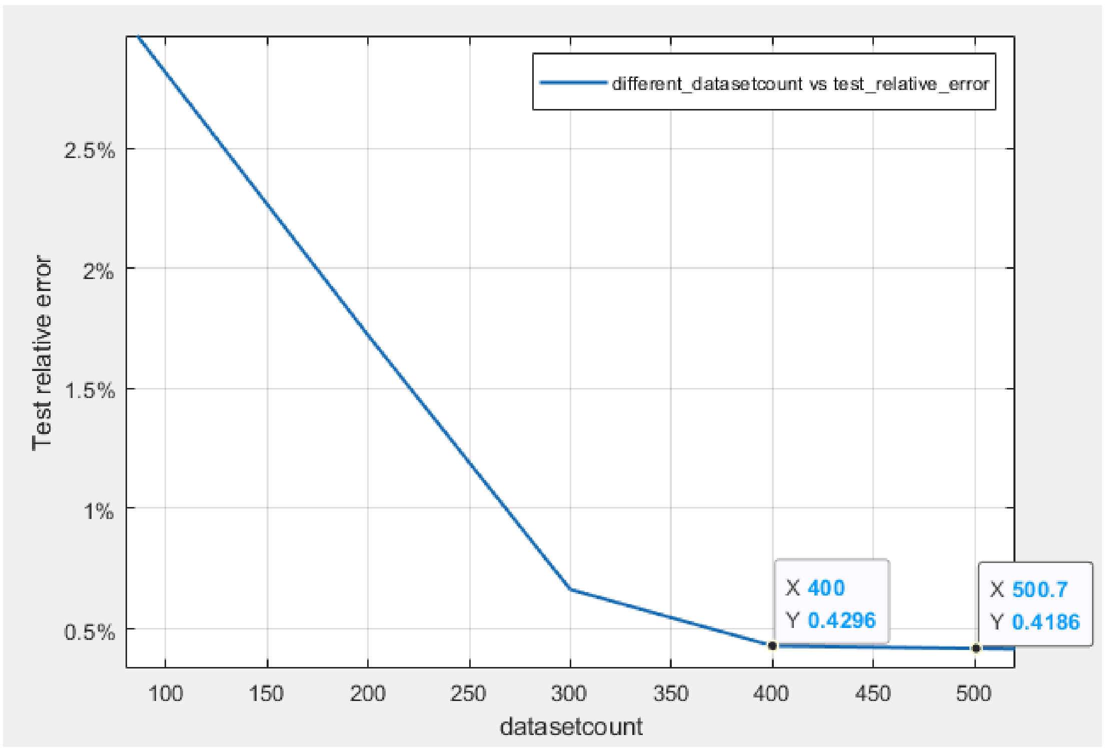


Figure 5-18: Relative error between the measured and predicted value vs. the size of the dataset.

5.5 Adding Noise To The Dataset

It is worth mentioning that our data are all from FEA and these are entered manually at the input side perfectly and without any errors. However, there are many errors in real experiments, especially when measuring the thickness of PCBs and chips and the solder joint volume and the manufacturing tolerance of the PCBs and chips. These common errors can greatly affect the accuracy of ML prediction models. To prevent this from happening, it's necessary to artificially add Gaussian noise to the thickness of the PCB and chip as well as the solder joint volume.

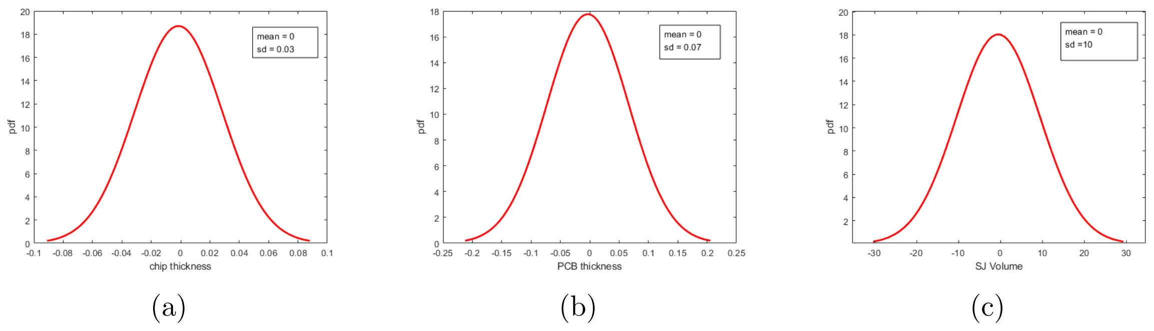


Figure 5-19: Gaussian noise. (a)Gaussian noise of chip thickness, mean is 0, standard deviation(SD) is 0.03. (b)Gaussian noise of PCB thickness, mean is 0, standard deviation is 0.07. (c)Gaussian noise of solder joint volume, mean is 0, standard deviation is 10.

Fig.5-19 illustrates the noise distribution on the thickness of PCBs and chips and the solder joint volume. Next, 300 sets of data were selected from 500 simulated data and three types of noise were added to each set, and then 50 sets of data were selected and three types of noise were added simultaneously, while ensuring that the other feature values and outputs were kept constant. Fig.5-20 shows the variation between these 3 feature values with and without noise.

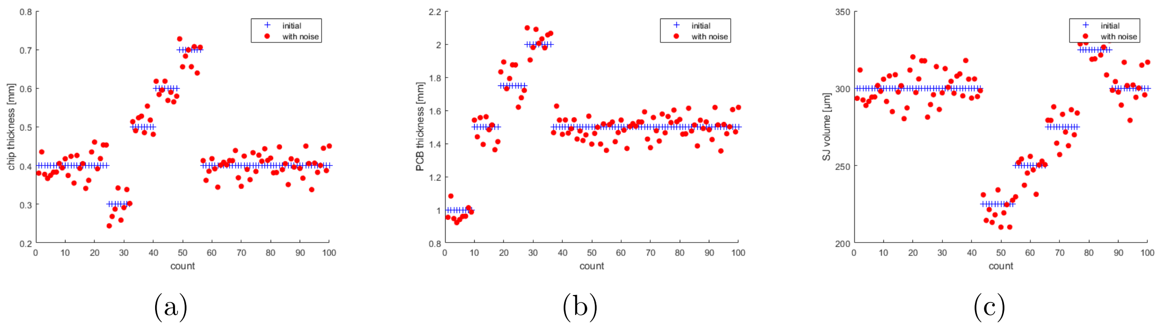


Figure 5-20: Variation between values with and without noise, where the blue plus sign is the value without noise and the red dot is the value with noise. (a)Variation between chip thickness with and without noise. (b)Variation between PCB thickness with and without noise. (c)Variation between solder joint volume with and without noise.

5.6 Model Training

In general, if the dataset is large enough, e.g. tens or even hundreds of thousands, the dataset will be divided before training the network. 70% of the total dataset is used as the training dataset, 15% as the validation dataset, and the remaining 15% for testing the trained model. Moreover, the training dataset is used to train the model and update the parameters(weights and bias), while the validation dataset, which is not involved in training, is used to verify whether the parameters updated at each epoch make the Loss of the data in the validation dataset smaller. If the updated hyperparameters obtained from an epoch make the Loss of the validation dataset larger than the previous one, training is stopped to prevent overfitting. The test dataset is also not involved in training; it is used to test the accuracy of the finalized model.

However, for smaller datasets, it is more preferable to train each data. So here the data is not divided and all the data is used as training datasets. Fig.5-21 shows the training data set with a total of 850 data. These data contain perfectly simulated data as well as data with noise.

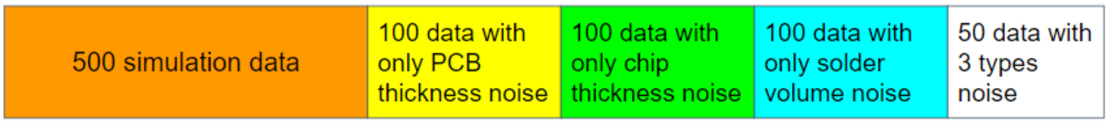

500 simulation data
100 data with only PCB thickness noise
100 data with only chip thickness noise
100 data with only solder volume noise
50 data with 3 types noise

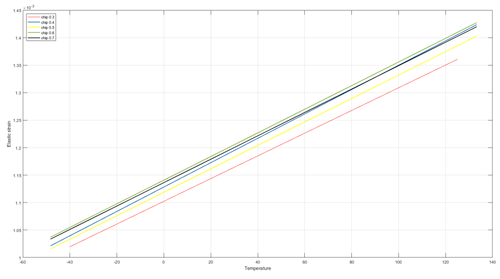
Figure 5-21: Training dataset.

Due to the lack of validation dataset, regarding the optimization algorithm for back propagation, Bayesian regularization[27] is chosen, and its regularization method can effectively prevent overfitting.

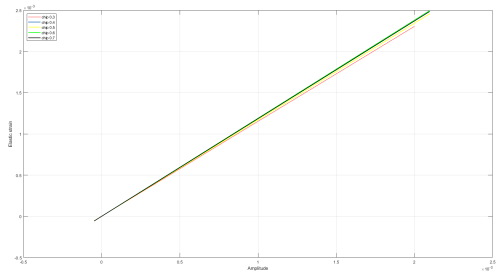
6 Results and Evaluation

6.1 Impact Of Features On Simulation Results

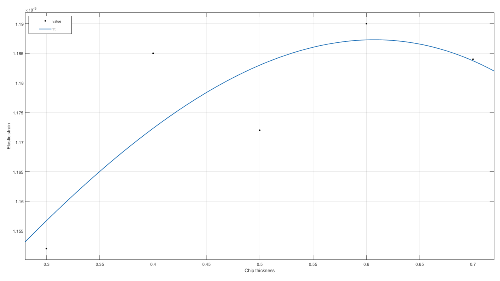
In the previous chapter we have introduced the features and outputs in the dataset. Tab.5.2 shows the values taken for these 6 different features. The range of values for each of these features is [-40°C, 125°C], [0.05mm, 2mm], [1mm, 2mm], [0.3mm, 0.7mm], [225μm, 325μm], [10, 25]. To investigate the effect of these six features on the simulation results, the control variable method is used, i.e., one of the features is changed and the other features maintain their standard values (temperature: 25°C, amplitude: 1mm, PCB thickness: 1.5mm, chip thickness: 0.4mm, solder diameter: 300μm, solder count: 25). The results are shown in Fig.6-22, Fig.6-23, Fig.6-24, Fig.6-25. It is clear that the equivalent elastic strain increases with increasing temperature or amplitude for different geometric values, and this relationship is linear. However, for the same temperature and amplitude, different geometric values have different effects on the simulation results. For example, the fitted curve of temperature, amplitude and equivalent elastic strain resembles a quadratic function with a maximum at a point. The fitted curve of solder joint volume (diameter) and equivalent elastic deformation is approximately linear, and the equivalent elastic deformation decreases significantly as the solder joint volume increases. The most interesting is the fitted curve of solder count versus equivalent elastic strain, which converges after the solder count is greater than 17.



(a)

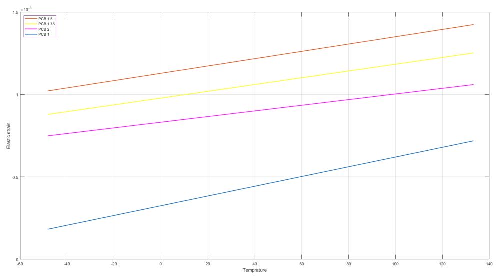


(b)

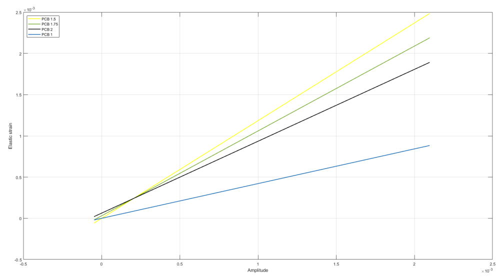


(c)

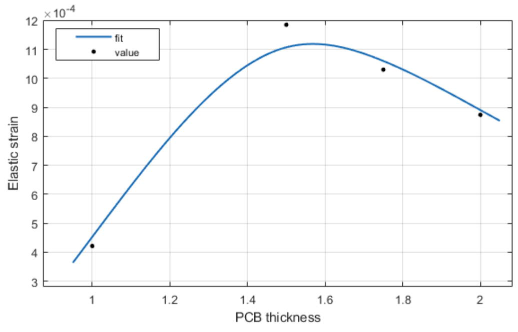
Figure 6-22: (a)Temperature vs Equivalent Elastic strain with different chip thickness. (b)Amplitude vs Equivalent Elastic strain with different chip thickness. (c)Chip thickness vs Equivalent Elastic strain.



(a)

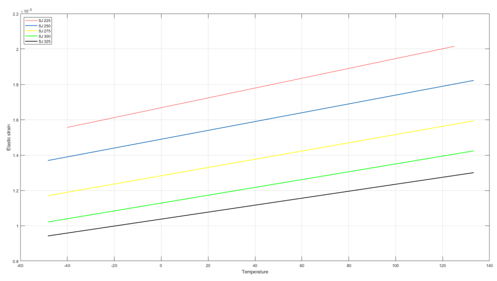


(b)

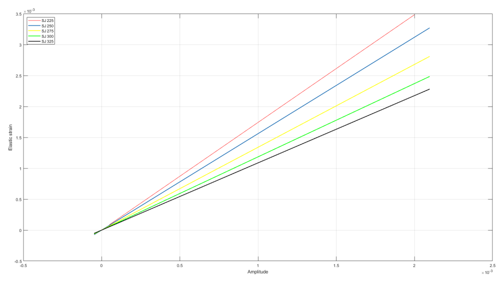


(c)

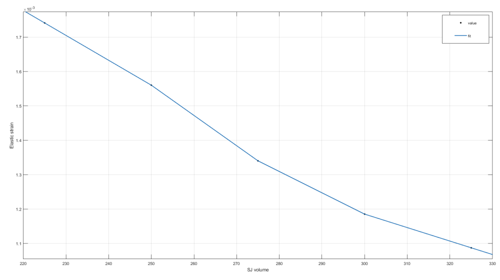
Figure 6-23: (a)Temperature vs Equivalent Elastic strain with different PCB thickness. (b)Amplitude vs Equivalent Elastic strain with different PCB thickness. (c)PCB thickness vs Equivalent Elastic strain.



(a)

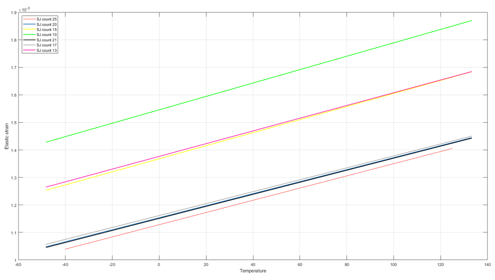


(b)

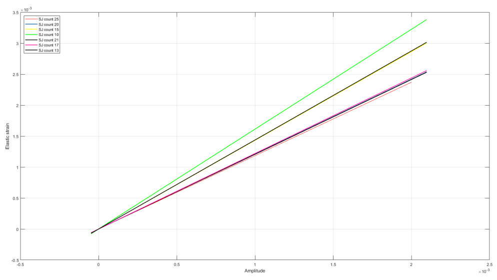


(c)

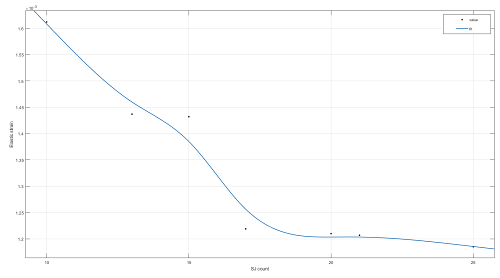
Figure 6-24: (a)Temperature vs Equivalent Elastic strain with different SJ volume. (b)Amplitude vs Equivalent Elastic strain with different SJ volume. (c)SJ volume vs Equivalent Elastic strain.



(a)



(b)



(c)

Figure 6-25: (a)Temperature vs Equivalent Elastic strain with different SJ count. (b)Amplitude vs Equivalent Elastic strain with different SJ count. (c)SJ count vs Equivalent Elastic strain.

6.2 Performance Evaluation Of ML Model

The ML model is trained over 500 epochs. The resultant loss after each pass through the dataset is recorded, as shown in Fig.6-26. The final training loss is found to converge at the 492nd epoch with a result of 0.077387%.

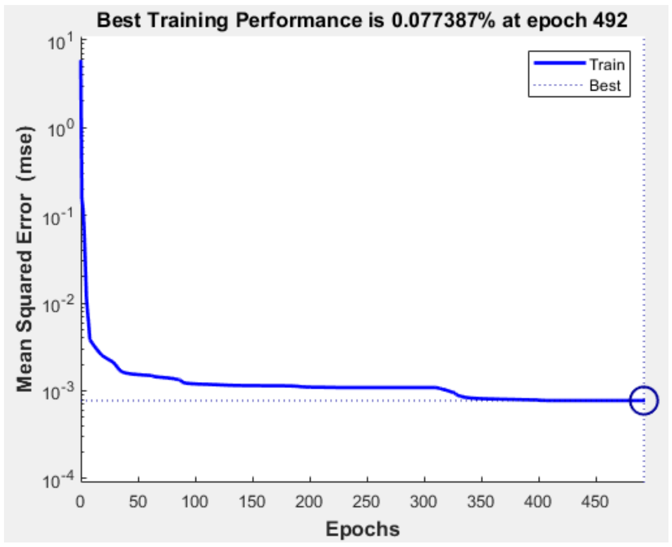


Figure 6-26: ML loss versus epochs of training.

In order to evaluate the performance of the network, it is necessary to establish an estimate of the accuracy of the ML model. Fig.6-27 gives a plot of the predicted equivalent elastic strain values versus the measured values for the ML model of the solder joint. The relative error between the measured and predicted values is 1.2296%.

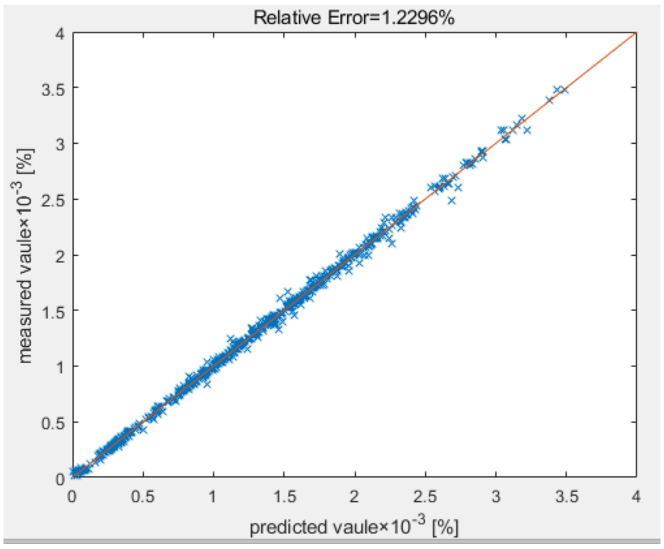


Figure 6-27: Predicted versus measured values for the neural network training dataset.

6.3 Comparing Simulation Results With ML Model Results

To compare simulation results with ML model results, 10 sets of data were selected from 500 sets of simulated data, as shown in Tab.6.1. Gaussian noise of PCB, chip thickness and solder joint volume are randomly added to these 10 sets of data as shown in Tab.6.2. The final relative error between the predicted and measured values of these 10 sets of data is 3.4155%, as shown in Fig.6-28.

Table 6.1: Raw dataset : 10 sets of data from 500 sets of simulated data.

features / count	Temperature	Amplitude	PCB thickness	Chip thickness	Solder joint volume	Solder joint count
①	25	0.0015	2	0.4	300	25
②	25	0.002	1.5	0.4	250	25
③	75	0.001	1.5	0.4	325	25
④	-40	0.001	1.5	0.4	325	25
⑤	125	0.001	1.75	0.4	300	25
⑥	100	0.00125	1.5	0.4	250	25
⑦	-25	0.00175	2	0.4	300	25
⑧	0	0.00075	1.75	0.4	300	25
⑨	50	0.00125	1.5	0.4	275	25
⑩	75	0.00125	1.5	0.4	275	25

Table 6.2: Test dataset : 10 sets of data from 500 sets of simulated data with noise. The values marked in color are the values after adding noise.

features / count	Temperature	Amplitude	PCB thickness	Chip thickness	Solder joint volume	Solder joint count
①	25	0.0015	1.94	0.37	292	25
②	25	0.002	1.38	0.42	251	25
③	75	0.001	1.67	0.36	330	25
④	-40	0.001	1.62	0.33	332	25
⑤	125	0.001	1.85	0.37	294	25
⑥	100	0.00125	1.55	0.38	261	25
⑦	-25	0.00175	2.17	0.39	291	25
⑧	0	0.00075	1.77	0.43	295	25
⑨	50	0.00125	1.57	0.39	271	25
⑩	75	0.00125	1.61	0.68	2288	25

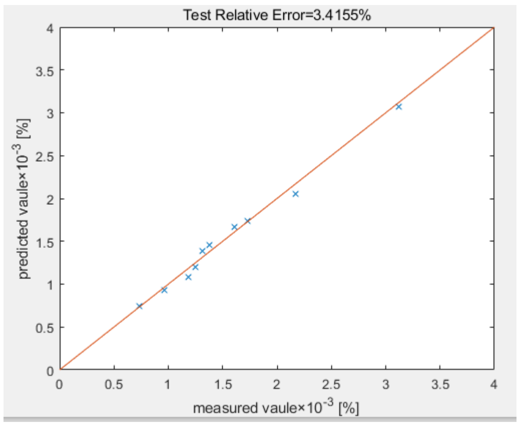


Figure 6-28: Predicted versus measured values for the neural network test dataset.

7 Discussion

In this research, we propose a machine learning model to predict the reliability of solder joints quickly and accurately, and the model is resistant to noise interference. Compared to traditional statistical methods, ML models are more accurate and faster. Moreover, from the data obtained from FEA, the temperature and amplitude have a greater impact on the simulation results than the geometry of the PCB, chip, and solder joints.

However, comparing with other literature, our model trains only one material(SAC305) of the solder joint and therefore ignores the chemical composition and physical properties of other materials. These have a significant impact on the simulation results because the solder joints react chemically with the PCB and the chip during thermal cycling, generating different compounds that can lead to CTE changes and affect the simulation results. Therefore, there will be significant limitations in practical applications. Moreover, as an output simulation result, we have selected only the equivalent elastic strain on the first solder joint. In fact, the aging and damage of the solder joint is caused by a combination of creep and fatigue. In addition among some external factors that affect the reliability of solder joints, we only modeled specific temperatures and amplitudes, but neglected some important properties in the thermal cycle, such as dwelling temperature, dwelling time, and rate of temperature change. However adding all the above mentioned features, such as properties of the solder joint material, parameters in the thermal cycle, into the dataset is a long and complex task, which is difficult to accomplish in a short time without additional data provided.

The most important thing is, as mentioned in the 'data collection' section, due to the limitations of the simulation software Ansys, we cannot consider all combinations of PCB, chip thickness and solder joint volume. As shown in Fig.7-29, the blue points are combinations that are present in the dataset, and the red points are missing combinations. This means that using our model to predict the combination of these red points leads to inaccurate predictions.

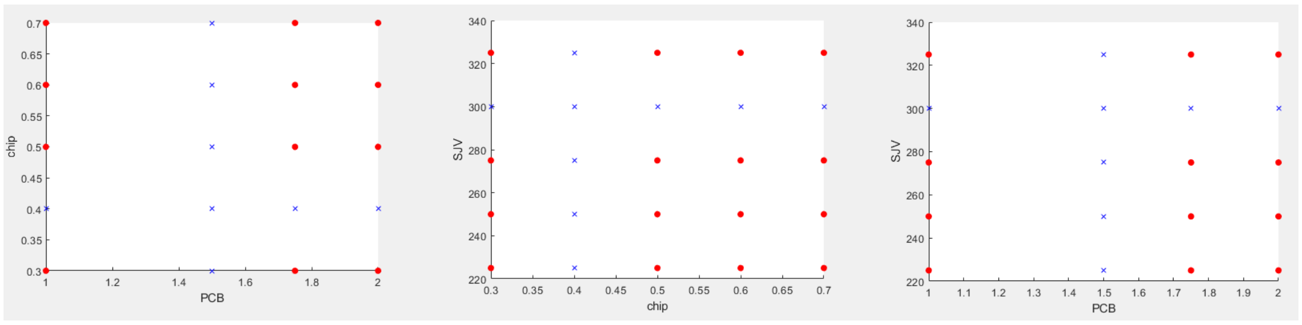


Figure 7-29: Possible combinations of PCB, chip thickness and solder joint volume.

.

8 Conclusion And Outlook

Overall, this paper presents an ML method to predict the reliability of solder joints in isothermal vibration tests. Extensive literature was reviewed to determine the appropriate input features, i.e. temperature, amplitude and geometry values. Then 500 simulation data obtained from FEA was collected and Gaussian noise was artificially added to prevent inaccurate prediction results due to measurement errors and manufacturing tolerance of chips and PCBs in the experiments. Next, build the neural network in Matlab and develop the ML infrastructure and the appropriate size of the training dataset. Finally, it was determined that the neural network has 2 hidden layers with 5 neurons on each hidden layer. 500 sets of simulated data and 350 sets of data with added noise were used as the training dataset. The weights and biases are updated using a Bayesian regularized BP optimization algorithm to prevent overfitting due to the absence of a validation dataset. As a result of the training, the final relative error between the predicted and measured values is obtained as 1.2296%. 10 sets of simulation results were also selected and compared with the ML model results. The final relative error is obtained as 3.4155%.

In the further work, a mixture of simulation and experimental data can be considered to increase the amount of datasets to fill the missing combinations. Alternatively, create models on Ansys containing the boundary combinations in Fig.7-29 and then use interpolation to fill in all the missing combinations.

9 Appendix

9.1 500 Data From FEA

Data download address

9.2 Code: Investigate The Effect Of Features On The Simulation Results

```
%{
This section of the code aims to investigate the effect of these
six features on the simulation results , the control variable method
is used , i.e., one of the features is changed and the other features
maintain their standard values.
The drawing tool is MATLAB's own cftool
%}

load('data')

%PCB thickness extract
%PCB thickness index
PCB_a1 = find(data(:,2)==0.001&data(:,3)==1&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);
PCB_b1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==25);
PCB_c1 = find(data(:,2)==0.001&data(:,3)==1.75&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==25);
PCB_d1 = find(data(:,2)==0.001&data(:,3)==2&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);

%temperature for different PCB thickness
Ptem_a = data(PCB_a1',1);
Ptem_b = data(PCB_b1',1);
Ptem_c = data(PCB_c1',1);
Ptem_d = data(PCB_d1',1);

%result for different PCB thickness with different tempreture
Pees_a1 = data(PCB_a1',7);
Pees_b1 = data(PCB_b1',7);
Pees_c1 = data(PCB_c1',7);
Pees_d1 = data(PCB_d1',7);

% PCB thickness index
PCB_a2 = find(data(:,1)==25&data(:,3)==1&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);
PCB_b2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);
PCB_c2 = find(data(:,1)==25&data(:,3)==1.75&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);
```

```
PCB_d2 = find(data(:,1)==25&data(:,3)==2&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);

% amplitude for different PCB thickness
Pamp_a = data(PCB_a2',2);
Pamp_b = data(PCB_b2',2);
Pamp_c = data(PCB_c2',2);
Pamp_d = data(PCB_d2',2);

% result for different PCB thickness with different amplitude
Pees_a2 = data(PCB_a2',7);
Pees_b2 = data(PCB_b2',7);
Pees_c2 = data(PCB_c2',7);
Pees_d2 = data(PCB_d2',7);

% PCB thickness index
PCB_thickness_index = find(data(:,2)==0.001&data(:,1)==25&data(:,4)
    ==0.4&data(:,5)==300&data(:,6)==25);

%value of PCB thickness
PCB_thickness = data(PCB_thickness_index',3);

%result for different PCB thickness with same amplitude and
    temperature
ess_PCB_thickness = data(PCB_thickness_index',7);

% same as the PCB extract
chip_a1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.3&data
    (:,5)==300&data(:,6)==25);
chip_b1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==25);
chip_c1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.5&data
    (:,5)==300&data(:,6)==25);
chip_d1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.6&data
    (:,5)==300&data(:,6)==25);
chip_e1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.7&data
    (:,5)==300&data(:,6)==25);

Ctem_a = data(chip_a1',1);
Ctem_b = data(chip_b1',1);
Ctem_c = data(chip_c1',1);
Ctem_d = data(chip_d1',1);
Ctem_e = data(chip_e1',1);

Cees_a1 = data(chip_a1',7);
Cees_b1 = data(chip_b1',7);
Cees_c1 = data(chip_c1',7);
Cees_d1 = data(chip_d1',7);
Cees_e1 = data(chip_e1',7);
```

```
chip_a2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.3&data(:,5)
    ==300&data(:,6)==25);
chip_b2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);
chip_c2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.5&data(:,5)
    ==300&data(:,6)==25);
chip_d2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.6&data(:,5)
    ==300&data(:,6)==25);
chip_e2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.7&data(:,5)
    ==300&data(:,6)==25);

Camp_a = data(chip_a2',2);
Camp_b = data(chip_b2',2);
Camp_c = data(chip_c2',2);
Camp_d = data(chip_d2',2);
Camp_e = data(chip_e2',2);

Cees_a2 = data(chip_a2',7);
Cees_b2 = data(chip_b2',7);
Cees_c2 = data(chip_c2',7);
Cees_d2 = data(chip_d2',7);
Cees_e2 = data(chip_e2',7);

chip_thickness_index = find(data(:,2)==0.001&data(:,1)==25&data(:,3)
    ==1.5&data(:,5)==300&data(:,6)==25);
chip_thickness = data(chip_thickness_index',4);
ess_chip_thickness = data(chip_thickness_index',7);

SJV_a1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==225&data(:,6)==25);
SJV_b1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==250&data(:,6)==25);
SJV_c1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==275&data(:,6)==25);
SJV_d1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==25);
SJV_e1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==325&data(:,6)==25);

Vtem_a = data(SJV_a1',1);
Vtem_b = data(SJV_b1',1);
Vtem_c = data(SJV_c1',1);
Vtem_d = data(SJV_d1',1);
Vtem_e = data(SJV_e1',1);

Vees_a1 = data(SJV_a1',7);
Vees_b1 = data(SJV_b1',7);
Vees_c1 = data(SJV_c1',7);
```

```
Vees_d1 = data(SJV_d1',7);
Vees_e1 = data(SJV_e1',7);

SJV_a2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==225&data(:,6)==25);
SJV_b2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==250&data(:,6)==25);
SJV_c2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==275&data(:,6)==25);
SJV_d2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);
SJV_e2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==325&data(:,6)==25);

Vamp_a = data(SJV_a2',2);
Vamp_b = data(SJV_b2',2);
Vamp_c = data(SJV_c2',2);
Vamp_d = data(SJV_d2',2);
Vamp_e = data(SJV_e2',2);

Vees_a2 = data(SJV_a2',7);
Vees_b2 = data(SJV_b2',7);
Vees_c2 = data(SJV_c2',7);
Vees_d2 = data(SJV_d2',7);
Vees_e2 = data(SJV_e2',7);

SJV_index = find(data(:,2)==0.001&data(:,1)==25&data(:,3)==1.5&data
    (:,4)==0.4&data(:,6)==25);
SJV = data(SJV_index',5);
ess_SJV = data(SJV_index',7);

SJC_a1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==10);
SJC_b1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==13);
SJC_c1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==15);
SJC_d1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==17);
SJC_e1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==20);
SJC_f1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==21);
SJC_g1 = find(data(:,2)==0.001&data(:,3)==1.5&data(:,4)==0.4&data
    (:,5)==300&data(:,6)==25);

SCtem_a = data(SJC_a1',1);
SCtem_b = data(SJC_b1',1);
SCtem_c = data(SJC_c1',1);
```

```
SCtem_d = data(SJC_d1',1);
SCtem_e = data(SJC_e1',1);
SCtem_f = data(SJC_f1',1);
SCtem_g = data(SJC_g1',1);

SCees_a1 = data(SJC_a1',7);
SCees_b1 = data(SJC_b1',7);
SCees_c1 = data(SJC_c1',7);
SCees_d1 = data(SJC_d1',7);
SCees_e1 = data(SJC_e1',7);
SCees_f1 = data(SJC_f1',7);
SCees_g1 = data(SJC_g1',7);

SJC_a2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==10);
SJC_b2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==13);
SJC_c2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==15);
SJC_d2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==17);
SJC_e2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==20);
SJC_f2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==21);
SJC_g2 = find(data(:,1)==25&data(:,3)==1.5&data(:,4)==0.4&data(:,5)
    ==300&data(:,6)==25);

SCamp_a = data(SJC_a2',2);
SCamp_b = data(SJC_b2',2);
SCamp_c = data(SJC_c2',2);
SCamp_d = data(SJC_d2',2);
SCamp_e = data(SJC_e2',2);
SCamp_f = data(SJC_f2',2);
SCamp_g = data(SJC_g2',2);

SCees_a2 = data(SJC_a2',7);
SCees_b2 = data(SJC_b2',7);
SCees_c2 = data(SJC_c2',7);
SCees_d2 = data(SJC_d2',7);
SCees_e2 = data(SJC_e2',7);
SCees_f2 = data(SJC_f2',7);
SCees_g2 = data(SJC_g2',7);

SJC_index = find(data(:,2)==0.001&data(:,1)==25&data(:,3)==1.5&data
    (:,4)==0.4&data(:,5)==300);
SJC = data(SJC_index',6);
ess_SJC = data(SJC_index',7);
```

9.3 Code: Adding Noise To The Dataset

```
%{
The code in this section mainly selects suitable data (350 groups)
from 500 groups of data to add noise, uses the function normrnd()
to randomly select noise from the Gaussian distribution with
specified mean and std to add to the data, and draws a scatter plot
to show the change of the data after adding noise.
Variable Meaning:
PCB, chip, SJV, mix: Data to be added to the noise
R1, R2, R3, mix_R1, mix_R2, mix_R3: Gaussian noise
noise_PCB, noise_chip, noise_SJV, noise_mix: Data after adding
   Gaussian noise
%}
load('data') %Importing Data

%Select the data to be added to the noise
PCB = data(1:100, :);
chip = data(171:270, :);
SJV = data(351:450, :);
mix = data([82:93 196:202 318:348 ],:);

% Set random number seed
rng(1);

% Adding PCB Noise
R1 = normrnd(0, 0.07, 1, 100); %Gaussian noise, specifying its mean
   and std and quantity
PCB_vakue = PCB(:,3);
PCB_vakue_with_noise = PCB_vakue + R1';
noise_PCB = [PCB(:,1:2) PCB_vakue_with_noise PCB(:,4:7)];
%save
save('noise_PCB');

%plot scatter
figure(1)
x = [1:1:100];
scatter(x,PCB_vakue,[],'b','+');hold on;
scatter(x,PCB_vakue_with_noise,[],'r','o','filled');
hold off;
legend('initial','with noise');
xlabel('count');
ylabel('PCB thickness [mm]');

% Adding chip Noise
R2 = normrnd(0, 0.03, 1, 100);
chip_vakue = chip(:,4);
chip_vakue_with_noise = chip_vakue + R2';
noise_chip = [chip(:,1:3) chip_vakue_with_noise chip(:,5:7)];
```

```
save('noise_chip')

%plot scatter
figure(2)
x = [1:1:100];
scatter(x,chip_vakue,[],'b','+');hold on;
scatter(x,chip_vakue_with_noise,[],'r','o','filled');
hold off;
legend('initial','with noise');
xlabel('count');
ylabel('chip thickness [mm]');

% Adding SJV Noise
R3 = normrnd(0, 10, 1, 100);
SJV_vakue = SJV(:,5);
SJV_vakue_with_noise = SJV_vakue + R3';
noise_SJV = [SJV(:,1:4) SJV_vakue_with_noise SJV(:,6:7)];
save('noise_SJV')

%plot scatter
figure(3)
x = [1:1:100];
scatter(x,SJV_vakue,[],'b','+');hold on;
scatter(x,SJV_vakue_with_noise,[],'r','o','filled');
hold off;
legend('initial','with noise');
xlabel('count');
ylabel('SJ volume [ m ]');


% Adding mix Noise
mix_R1 = normrnd(0, 0.07, 1, 50);
mix_R2 = normrnd(0, 0.03, 1, 50);
mix_R3 = normrnd(0, 10, 1, 50);
mix_1 = mix(:,3);
mix_2 = mix(:,4);
mix_3 = mix(:,5);
mix_1_with_noise = mix_1 + mix_R1';
mix_2_with_noise = mix_2 + mix_R2';
mix_3_with_noise = mix_3 + mix_R3';
noise_mix = [mix(:,1:2) mix_1_with_noise mix_2_with_noise
   mix_3_with_noise mix(:,6:7)];
save('noise_mix')
```

9.4 Code: Build, Train And Evaluate The Neural Network.

```
%{
This section of code aims to build, train and evaluate the neural
network. First 500 sets of simulated data and 350 sets of data
with noise will be integrated into a new data set, then the inputs
and outputs will be extracted from these 850 sets of data and the
   inputs
will be normalized. The network is built using fitnet(), the
   parameters
of the network as well as the hyperparameters are determined,
the network is trained using train(), and finally the accuracy
and speed of the network is evaluated and plotted.
Variable Meaning:
data: 500 simulation data and 350 data with noise
X,Y: input and output
hiddenlayerSize, net.trainParam.lr :Hyperparameters
Train_Relative_Error : Metrics for neural network evaluation
%}

format long;
load('data') % Importing 500 simulation Data
load('noise_chip') % Importing 100 Data with chip noise
load('noise_mix') % Importing 50 Data with mix noise
load('noise_PCB') % Importing 100 Data with PCB noise
load('noise_SJV') % Importing 100 Data with SJV noise

% Set random number seed
rng(1);

% extract input and output from dataset
data = [data;noise_chip;noise_PCB;noise_SJV;noise_mix];
X = data(:,1:6);% input
Y = data(:,7); % output

% normalize the input in [0.2 0.8]
[Xn,Xs]= mapminmax(X',0.2,0.8);
xt = Xn;
%Increase the output so that the loss is not too
%small to cause the training to stop early.
yt = Y'*1000;

% build BP network framework
hiddenlayerSize = [5 5];%Number of neurons on the hidden layer
trainFcn = 'trainbr'; % BP function
net = fitnet(hiddenlayerSize, trainFcn);
net.divideFcn = ''; %No validation dataset and test dataset
net.trainParam.lr = 0.01; %set learning rate
```

```
% train the model
tic;%Timer start
[net,tr] = train(net, xt, yt);%train the model
tr.divideFcn = '' ;
toc;%Timer end



% evaluation and plot

yTrain = net(xt(:,tr.trainInd));%predicted value * 1000
yTrainTrue = yt(tr.trainInd); %measured value * 1000
yTrain1 = yTrain/1000; %predicted value
yTrainTrue1 = yTrainTrue/1000;%measured value
%relative error
Train_Relative_Error = abs(yTrain1 -yTrainTrue1)/yTrainTrue1 * 100;

grid on;
figure(1)
plot(yTrain,yTrainTrue,'x');hold on;
plot(0:4,0:4);hold off;
xlabel('predicted␣vaule 10^-^3␣[%]');
ylabel('measured␣vaule 10^-^3␣[%]');
text(1,3,'Train␣dataset');
title(['Train␣Relative␣Error=',num2str(Train_Relative_Error),'%']);

%save
save('net')
```

9.5 Code: Predict.

```
%{
This section of the code aims to use the trained ML model
to make predictions. The input feature values are the first
to enter, followed by normalization of the feature values,
and finally, the predicted output values are obtained.
Variable Meaning:
x_predict: features,
i.e.[temperature,amplitude,PCB thicknee,chip thckness,SJV,SJC]
y_predict: predicted value
%}
load('net')%Importing the neural network and its parameters

x_predict = [];%features
x_predict = mapminmax('apply',x_predict',Xs);
y_predict = sim(net,x_predict);

digits(4);%Rounded to 4 decimal places
y_predict = y_predict/1000 %predicted value
```

References

[1] Emeka H. Amalu and N.N. Ekere. “Modelling evaluation of Garofalo-Arrhenius creep relation for lead-free solder joints in surface mount electronic component assemblies”. In: *Journal of Manufacturing Systems* 39 (2016), pp. 9–23. ISSN: 0278-6125. DOI: `https://doi.org/10.1016/j.jmsy.2016.01.002`. URL: `https://www.sciencedirect.com/science/article/pii/S0278612516000030`.

[2] Babak Arfaei et al. “Reliability and failure mechanism of solder joints in thermal cycling tests”. In: *2013 IEEE 63rd Electronic Components and Technology Conference*. IEEE. 2013, pp. 976–985.

[3] P. Borgesen et al. “Solder joint reliability under realistic service conditions”. In: *Microelectronics Reliability* 53.9 (2013). European Symposium on Reliability of Electron Devices, Failure Physics and Analysis, pp. 1587–1591. ISSN: 0026-2714. DOI: `https://doi.org/10.1016/j.microrel.2013.07.091`. URL: `https://www.sciencedirect.com/science/article/pii/S0026271413002679`.

[4] Peter Borgesen et al. “Solder joint reliability under realistic service conditions”. In: *Microelectronics Reliability* 53.9-11 (2013), pp. 1587–1591.

[5] Preeti Chauhan et al. “Critical review of the Engelmaier model for solder joint creep fatigue reliability”. In: *IEEE Transactions on Components and Packaging Technologies* 32.3 (2009), pp. 693–700.

[6] F.X. Che and John H.L. Pang. “Vibration reliability test and finite element analysis for flip chip solder joints”. In: *Microelectronics Reliability* 49.7 (2009), pp. 754–760. ISSN: 0026-2714. DOI: `https://doi.org/10.1016/j.microrel.2009.03.022`. URL: `https://www.sciencedirect.com/science/article/pii/S0026271409000973`.

[7] Tzu-Chia Chen et al. “Application of machine learning in rapid analysis of solder joint geometry and type on thermomechanical useful lifetime of electronic components”. In: *Mechanics of Advanced Materials and Structures* (2021), pp. 1–9.

[8] Tzu-Chia Chen et al. “Estimation of Thermomechanical Fatigue Lifetime of Ball Grid Solder Joints in Electronic Devices Using a Machine Learning Approach”. In: *Journal of Electronic Materials* (2022), pp. 1–9.

[9] Kuo-Ning Chiang and Chang-An Yuan. “An overview of solder bump shape prediction algorithms with validations”. In: *IEEE transactions on advanced packaging* 24.2 (2001), pp. 158–162.

[10] Maurice N Collins, Jeff Punch, and Richard Coyle. “Surface finish effect on reliability of SAC 305 soldered chip resistors”. In: *Soldering & Surface Mount Technology* (2012).

[11] Andreas R Fix, Wolfgang Nüchter, and Jürgen Wilde. “Microstructural changes of lead-free solder joints during long-term ageing, thermal cycling and vibration fatigue”. In: *Soldering & Surface Mount Technology* (2008).

[12] Stephen M Heinrich et al. “Analytical expressions for shear and axial joint deformations in area-array assemblies due to global CTE mismatch”. In: *ASME International Mechanical Engineering Congress and Exposition*. Vol. 36487. 2002, pp. 485–497.

[13] jeffheaton. *The Number of Hidden Layers*. 2008. URL: `https://web.archive.org/web/20140721050413/http://www.heatonresearch.com/node/707`.

[14] Marion Branch Kelly, Antony Kirubanandham, and Nikhilesh Chawla. "Mechanisms of thermal cycling damage in polycrystalline Sn-rich solder joints". In: *Materials Science and Engineering: A* 771 (2020), p. 138614.

[15] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[16] J Liang et al. "Effects of load and thermal conditions on Pb-free solder joint reliability". In: *Journal of Electronic Materials* 33.12 (2004), pp. 1507–1515.

[17] Xu Long et al. "Parameterized Anand constitutive model under a wide range of temperature and strain rate: experimental and theoretical studies". In: *Journal of Materials Science* 55 (2020), pp. 10811–10823.

[18] TT Nguyen, D Yu, and SB Park. "Characterizing the mechanical properties of actual SAC105, SAC305, and SAC405 solder joints by digital image correlation". In: *Journal of Electronic Materials* 40.6 (2011), pp. 1409–1415.

[19] Guangyuan Ren et al. "Pb-free solder—microstructural, material reliability, and failure relationships". In: 2020.

[20] Vahid Samavatian, Hossein Iman-Eini, and Yvan Avenas. "An efficient online time-temperature-dependent creep-fatigue rainflow counting algorithm". In: *International Journal of Fatigue* 116 (2018), pp. 284–292. ISSN: 0142-1123. DOI: `https://doi.org/10.1016/j.ijfatigue.2018.06.037`. URL: `https://www.sciencedirect.com/science/article/pii/S0142112318302718`.

[21] Vahid Samavatian et al. "Correlation-driven machine learning for accelerated reliability assessment of solder joints in electronics". In: *Scientific reports* 10.1 (2020), pp. 1–14.

[22] Vahid Samavatian et al. "Iterative Machine Learning-Aided Framework Bridges Between Fatigue and Creep Damages in Solder Interconnections". In: *IEEE Transactions on Components, Packaging and Manufacturing Technology* 12.2 (2021), pp. 349–358.

[23] Dhafer Abdulameer Shnawah, Mohd Faizul Mohd Sabri, and Irfan Anjum Badruddin. "A review on thermal cycling and drop impact reliability of SAC solder joint in portable electronic products". In: *Microelectronics reliability* 52.1 (2012), pp. 90–99.

[24] Jianhao Wang et al. "The reliability of lead-free solder joint subjected to special environment: a review". In: *Journal of Materials Science: Materials in Electronics* 30.10 (2019), pp. 9065–9086.

[25] Jie Xiong, San-Qiang Shi, and Tong-Yi Zhang. "A machine-learning approach to predicting and understanding the properties of amorphous metallic alloys". In: *Materials Design* 187 (2020), p. 108378. ISSN: 0264-1275. DOI: `https://doi.org/10.1016/j.matdes.2019.108378`. URL: `https://www.sciencedirect.com/science/article/pii/S0264127519308160`.

[26] Sung Yi and Robert Jones. "Machine learning framework for predicting reliability of solder joints". In: *Soldering & Surface Mount Technology* 32.2 (2019), pp. 82–92.

[27] Zhao Yue, Zhao Songzheng, and Liu Tianshi. "Bayesian regularization BP Neural Network model for predicting oil-gas drilling cost". In: *2011 International Conference on Business Management and Electronic Information.* Vol. 2. 2011, pp. 483–487. DOI: `10.1109/ICBMEI.2011.5917952`.