Nonlinear Chen-Lee Chaotic System Based Deep Convolutional Generative Adversarial Nets for Chatter Diagnosis

Ping-Huan Kuo  
National Chung Cheng University

Po-Chien Luan  
National Chung Cheng University

Yung-Ruen Tseng  
National Chung Cheng University

Her-Terng Yau (pan1012@ms52.hinet.net)  
National Chung Cheng University  https://orcid.org/0000-0002-1187-1771

Research Article

Keywords: generative adversarial network, convolutional neural network, chatter, chaos theory, fractional differential masks

Posted Date: December 8th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1066378/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
Abstract — Chatter has a direct effect on the precision and life of machine tools and its detection is a crucial issue in all metal machining processes. Traditional methods focus on how to extract discriminative features to help identify chatter. Nowadays, deep learning models have shown an extraordinary ability to extract data features which are their necessary fuel. In this study deep learning models have been substituted for more traditional methods. Chatter data are rare and valuable because the collecting process is extremely difficult. To solve this practical problem an innovative training strategy has been proposed that is combined with a modified convolutional neural network and deep convolutional generative adversarial nets. This improves chatter detection and classification. Convolutional neural networks can be effective chatter classifiers, and adversarial networks can act as generators that produce more data. The convolutional neural networks were trained using original data as well as by forged data produced by the generator. Original training data were collected and preprocessed by the Chen-Lee chaotic system. The adversarial training process used these data to create the generator and the generator could produce enough data to compensate for the lack of training data. The experimental results were compared with without a data generator and data augmentation. The proposed method had an accuracy of 95.3% on leave-one-out cross-validation over ten runs and surpassed other methods and models. The forged data were also compared with original training data as well as data produced by augmentation. The distribution shows that forged data had similar quality and characteristics to the original data. The proposed training strategy provides a high-quality deep learning chatter detection model.

Index Terms — generative adversarial network, convolutional neural network, chatter, chaos theory, fractional differential masks

I. INTRODUCTION

Many new research domains are emerging during Industry 4.0. Some of these are the Internet of things (IoT), big data, cloud computing, and smart manufacturing. Smart manufacturing aims at the development of advanced techniques to improve the efficiency and quality of traditional manufacture. The high-quality sensors now available allow the collection of many kinds of different signals during a manufacturing process. These signals provide valuable data that reflects the physical state of machine tools and can be used to develop AI algorithms.

Chatter, the vibrations that sometimes arise between the cutting tools and the workpiece during machining, can result in unacceptable degradation of product quality [1]. Chatter can also damage both the cutting tools and the machine itself [2]. There are two types of chatter, primary and secondary. Primary chatter arises at low spindle rotational speed and secondary chatter at higher speeds. Chatter can also occur in robotic and automatic machining processes [3]. Sliding mode control is a commonly used approach to the alleviation of chatter [4, 5]. In [6], asymmetric stiffness control is proposed for chatter control. Accurate real-time chatter detection can prevent damage and excessive tool wear in costly machine tools. There are many kinds of methods proposed to detect chatter precisely and rapidly [7–9]. Signal processing is important for the detection of chatter. The signals collected during machining processes are often very noisy. Many of the traditional techniques such as wavelet packet decomposition [10, 11], Fourier transformation [12] and others are widely used. However, all these traditional methods focus on the extraction of discriminative signals. These techniques need abundant domain knowledge making them difficult to realize and even though chatter can easily be detected using signal processing it is not easy to adapt to different conditions. Each time a cutting tool or a workpiece is changed the system will produce a completely different set of chatter signals [13].

Machine learning provides a distinct and powerful way of finding a generalized and robust model for classification and logistic regression offers an intuitive route to binary classification [14]. The support vector machine (SVM) has been widely used for chatter identification [15–17], but deep learning now dominates the machine learning-based methods [18] and has been very successful in image processing [19, 20], speech processing [21], and many other applications. Deep learning methods are based on...
feature learning and many studies have been done that use neural networks to identify chatter. The multilayer perceptron (MLP) has been employed for chatter detection by using tool vibration data [22] and neural networks are often trained using preprocessed, rather than raw noisy data, because the extraction of well-organized features is easier. Convolutional neural network (CNN) is widely implemented to solve classification problems [23, 24]. Continuous wavelet transform can also be applied to preprocess data for training CNN [25, 26]. Recurrent neural networks (RNN) are good at dealing with time sequenced data. A long short-term memory neural network (LSTM) is a powerful RNN model, that allows a memory cell to consider historical states. LSTM has been used in several chatter detection studies [27, 28].

Data is a key player in deep learning. The amount and quality of data has a direct effect on the training model and even small amounts of data can result in problems like overfitting. Unfortunately, the collection of data is a strenuous and troublesome process and amassing large amounts of data is both inefficient and unnecessary and could also take months, or even years. Chatter only occurs at specific feed rates and cutting force and also depends on the manufacturing process and the kind of cutting tool used [29]. Chatter can cause serious damage to the cutters and even the machine tools themselves. This can make the collection of chatter data very expensive. There are several ways in which this problem can be overcome: Transfer learning is a common approach to model training which also uses small amounts of data. Data augmentation can produce variants of original data for image classification and will also produce a more generalized model.

In this study, an instinctive solution was adopted through a generative model which was used to produce data for training. A new framework was proposed to cope with this awkward problem and a data generator was used that produced enough data for training with a small amount of data.

This paper is organized as follows: Section II presents the overview of system architecture, Section III introduces the proposed method and its background, Section IV shows the experimental results, and conclusion and future work are discussed in Section V.

II. SYSTEM ARCHITECTURE

The chatter vibration data used in this study were collected from a Hybrid Sphere CNC Lathe model MC4200BL (TAIWAN MAC EDUCATIONAL Co Ltd). The raw vibration signals were produced by a KS943B-100 accelerometer (Metra Mess-und Frequenztechnik) using an external magnet attached to the spindle of the lathe. The output signal was collected by an NI 9234C series sound and vibration input module and an NI USB 9162 portable bus-powered USB carrier (National Instruments) for acceleration measurement, as shown in Fig. 1.

The chatter is captured by the accelerometer installed on the spindle of the lathe and the AD/DA digital capture card. Some of the resulting signal traces are shown in Fig. 2. Jian et al. [30] pointed out that a cutting depth of 4 mm induced cutting forces that were large enough to cause chatter. The chatter processing parameters and environment used in this study were the same, and the details are shown in TABLE 1.

| TABLE 1 | MACHINING PARAMETERS TO PRODUCE CHATTER |
|----------------|------------------------------------------|
| Cutting tool    | Mitsubishi Tungsten                      |
| Specifications  | Carbide turning insert                    |
| Work piece size (mm*mm) | S45C 30 *175                |
| Spindle speed (rpm) | 1500                                      |
| Feed rate (mm/rev) | 0.2                                       |
| Depth of cut (mm) | 4                                         |
| Sampling rate (Hz) | 25.6K                                    |
| Amount of data (groups/set) | 5000                                    |
In this section the details of the proposed method are described. First, data pre-processing based on chaos theory is discussed and explained. Secondly, an improved convolutional neural networks model is introduced. Thirdly, the design of the data generator will be discussed; and finally the proposed training framework will be described step-by-step.
A. Data Preprocessing

In chaotic synchronization control theory, adding a controller to a slave system allows the transient response of the slave system to keep up with the master chaotic system and the dynamic error between the master and the slave chaotic systems becomes smaller (the smaller-the-better). In this study the Chen-Lee chaotic system \[31\] has been used and the equation (1) is shown below:

\[
\begin{align*}
\dot{x} &= -yz + ax \\
\dot{y} &= xz + \beta y \\
\dot{z} &= \left(\frac{1}{3}\right) xy + yz
\end{align*}
\]  

(1)

A most important characteristic of any chaotic system is that a very small change of the control system input parameter will cause a great change in output. Through the controller, see equation (2), the dynamic error between the master and slave chaotic systems, see equation (3), forms a strange attractor to maintain the chaotic characteristics.

\[
\begin{align*}
u_1 &= -y_2 e_3 - y_3 e_2 \\
u_2 &= y_1 e_3 + y_3 e_1 \\
u_3 &= \frac{1}{3}(y_1 e_2 + y_2 e_1)
\end{align*}
\]  

(2)

\[
\begin{bmatrix}
\dot{e}_1 \\
\dot{e}_2 \\
\dot{e}_3
\end{bmatrix} =
\begin{bmatrix}
\alpha & 0 & 0 \\
0 & \beta & 0 \\
0 & 0 & \gamma
\end{bmatrix}
\begin{bmatrix}
e_1 \\
e_2 \\
e_3
\end{bmatrix}
+ \begin{bmatrix}
-e_2 e_3 \\
e_1 e_3 \\
\frac{1}{3} e_1 e_2
\end{bmatrix}
\]  

(3)

The fractional order differential form introduced in this study is the fractional G-L calculus model \[32\], equation (4).

\[
D^\sigma(x^m) \approx \frac{\Gamma(m + 1)}{\Gamma(m - \alpha + 1)} x^{m-\sigma}
\]  

(4)

It was assumed that the fractional order chaotic system could extract features and effectively suppress noise. Using a combination of fractional order differential arithmetic and the Chen-Lee chaotic system, the fractional order Chen-Lee chaotic dynamic error equation, as shown in equation (5), was found:

\[
\begin{bmatrix}
D^\sigma e_1 \\
D^\sigma e_2 \\
D^\sigma e_3
\end{bmatrix} \approx
\begin{bmatrix}
\alpha & 0 & 0 \\
0 & \beta & 0 \\
0 & 0 & \gamma
\end{bmatrix}
\begin{bmatrix}
e_1^{1-\sigma} \\
e_2^{1-\sigma} \\
e_3^{1-\sigma}
\end{bmatrix}
+ \begin{bmatrix}
-\Gamma(1)e_2 e_3 e_1^{\sigma} \\
\Gamma(1)e_1 e_3 e_2^{\sigma} \\
\Gamma(1)e_1 e_2 e_3^{\sigma}
\end{bmatrix}
\]  

(5)

\[
\begin{bmatrix}
\varphi_1[i + 1] \\
\varphi_2[i + 1] \\
\varphi_3[i + 1]
\end{bmatrix} \approx
\begin{bmatrix}
\alpha & 0 & 0 \\
0 & \beta & 0 \\
0 & 0 & \gamma
\end{bmatrix}
\begin{bmatrix}
e_1[i]^{1-\sigma} \\
e_2[i]^{1-\sigma} \\
e_3[i]^{1-\sigma}
\end{bmatrix}
+ \begin{bmatrix}
-\Gamma(1)e_2[i] e_3[i] e_1[i]^{\sigma} \\
\Gamma(1)e_1[i] e_3[i] e_2[i]^{\sigma} \\
\Gamma(1)e_1[i] e_2[i] e_3[i]^{\sigma}
\end{bmatrix}
\]  

(6)

The acceleration signals from a general turning operation and those from flutter turning were converted into two-dimensional images produced by fractional chaotic dynamic errors. The differences are more obvious than in those from one-dimensional acceleration signals as shown in Fig. 3. It was clear that classification using a neural network gave better results. According to past experience \[30\], when the fractional parameter \(\sigma\) was set to 0.8, the neural network performance was stable, see Fig. 4. Therefore, 0.8 was used as the fractional parameter in this study. The shape and distribution of images determines the representation of features. Images in color offered no meaningful information in these experiments, and multi-channel convolutional neural networks merely increased computation time, two-dimensional images were therefore processed in grey level, see Fig. 5.
Fig. 3. Fractional order chaotic dynamic error maps of E1E2. Those shown on the left are with chatter and those on the right are without chatter.

Fig. 4. Results of different order of the fractional order chaotic dynamic error map (E1E2)—(a) $v=0.2$, (b) $v=0.4$, (c) $v=0.6$, (d) $v=0.8$, and (e) $v=1.0$.

Table: E1E2 E1E3 E2E3

Fig. 5. Fractional-order chaotic dynamic error maps shown in grey levels.
B. Fractional Order Convolution Neural Network

CNNs are widely used in image recognition and convolution layers can extract hierarchical features of images using different filters. In this study, a fractional-order convolutional network (FOCNN) was proposed for the diagnosis of chatter. Of course, a lightweight model was needed for the implementation of a model for online chatter detection. A very deep structure such as ResNet [33] or DenseNet [34] has good performance but was too unwieldy. FO CNN, the design of which is based on fractional differential masks and CNN, was suitable for the purpose. Fractional differential masks have been widely used for image processing [35] and in this study they have been designed to work as kernels within the convolutional layers. These kernels inherit the benefits of fractional differential masks such as immunity to noise and the enhancement of specified features. According to the Grünwald–Letnikov definition [36], fractional derivatives can be described as follows:

\[
\frac{d^v s(t)}{dt^v} \approx s(t) + (-v)s(t - 1) + \frac{(-v)(-v + 1)}{2} s(t - 2) + \cdots + \frac{\Gamma(-v + 1)}{n! \Gamma(-v + n + 1)} s(t - n)
\]

where \( s(t) \) is a one-dimensional signal, \( s(x, y) \) is two-dimensional data, and \( v \) is a non-integer parameter of fractional order. In (8) and (9), \( \epsilon_x^v s(x, y) \) and \( \epsilon_y^v s(x, y) \) are the corresponding approximate errors of the \( x \)- and \( y \)-axes. The first three terms of (8) and (9) were used to design 12 fractional kernels as shown in Fig. 6. The three terms are arranged in different directions. The direction of the nonzero parameters of these masks indicate the focus of the directional features of the mask, so that the filters can capture the features of the different directions. The pre-designed kernels can reduce the CNN parameters and accelerate the convergence rate. The parameter \( v \) will be updated through a gradient-based optimizer and back-propagation.

\[
\frac{\partial^v s(x, y)}{\partial x^v} \approx s(x, y) + (-v)s(x - 1, y) + \frac{(-v)(-v + 1)}{2} s(x - 2, y) + \epsilon_x^v s(x, y)
\]

\[
\frac{\partial^v s(x, y)}{\partial y^v} \approx s(x, y) + (-v)s(x, y - 1) + \frac{(-v)(-v + 1)}{2} s(x, y - 2) + \epsilon_y^v s(x, y)
\]

The diagram in Fig. 7 is a representation of the structure of FOCNN. There are few hidden layers and there are only 545 parameters and this satisfies the qualification of FOCNN as a lightweight model. Twelve fractional kernels are implemented in the fractional-order convolutional layer, each of which is divided into four more kernels with different trainable parameters \( v \). This makes a total of 48 kernels in the fractional-order convolutional layer. Fractional kernels of the same type are marked in the same color, as shown in Fig. 7 and an average layer averages 48 feature maps. A max-pooling layer, a convolutional layer with eight
kernels, a max-pooling layer, and a fully connected layer account for the subsequent network. Dropout was applied after the flatten layers that some noise was added to prevent overfitting. The dropout rate was set to 20% as determined by the experiments.

C. Data Generator

The selection of a data generator played a key role in this work. A powerful generator is needed to create reliable and diverse data to compensate for the data deficiency. The Generative Adversarial Network (GAN) proposed by Goodfellow et al. in 2014 [37] is one of the most popular generative models used in recent years. The invention of GAN helped usher in the revolution of generative models. Many studies have shown that GAN is capable of generating high-quality images [38] and the original GAN framework was used in this study. The datasets were not very complicated, so it was not necessary to use advanced GAN. The basic idea of GAN is introduced as follows.

The adversarial training framework simultaneously trains a generator and a discriminator. The generator \( G \) can be viewed as a painter attempting to imitate the appearance of real data. The discriminator \( D \) serves as an appraiser to tell whether the data is real or not. During the training process, \( G \) attempts to deceive the discriminator, and \( D \) tries to discriminate between real data and data generated by the generator. In other words, the generator seeks a distribution \( p_{z} \) to approximate the distribution of training data \( p_{data} \). The procedures can be expressed as a minmax function as below:

\[
\min_{G} \max_{D} V(D, G) = E_{x \sim p_{data}}[\log D(x)] + E_{z \sim p_{z}}[\log(1 - D(G(z)))]
\]

(10)

where \( V \) is the objective function, \( D \) is the function of the discriminator, \( G \) is the function of the generator, \( x \) is the data sampling from real data, \( z \) is noise sampling from a specific probability distribution \( p_z \). Giving the real data a high score allows \( D \) to correctly recognize real data, so it is necessary to maximize the first term. \( G \) aims to create fake data that will deceive \( D \), so the second term needs to be minimized.

To implement the GAN model in the program, the training process can be divided into two simple steps: in the first step, \( G \) is fixed, and \( D \) is trained to separate real data and data generated by \( G \); in the second step, \( D \) is fixed, and \( G \) is trained to generate data that misleads \( D \) to classify the fake data as real data. The two steps iteratively train \( D \) and \( G \). Finally, \( G \) can approximate the distribution of training data, and \( D \) cannot tell the difference between training data and data generated by \( G \).

The suggestions of deep convolutional generative adversarial networks (DCGAN) were adopted for the design of the GAN model [39] used in this study. The network structure is described in TABLE 2. \( G \) and \( D \) were connected with each other in the same network and marked in the table. According to the guidelines of DCGAN, the first layer of \( G \) and the last layer of \( D \) were designated as fully connected layers. Fractional-strided convolutions (ConvTranspose2D) were used for up-sampling data in the generator. In the discriminator, down-sampling was accomplished by strided convolutions (Conv2D). Different numbers of filters were set for each convolutional layer. Batch normalization was also introduced in the generator to prevent gradient vanishing and accelerate the convergence rate. ReLU [40] was chosen as the activation function in \( G \) and Leaky ReLU was implemented in \( D \) [41]. RMSProp was chosen as the optimizer to cope with the complex error surface and the complete training procedures of the GAN model is shown in Algorithm 1.

```
Algorithm 1: Adversarial training

Input: \( X \leftarrow \) training data;
\( G \leftarrow \) generator
\( D \leftarrow \) discriminator
\( K \leftarrow \) training iterations

Output: \( G \);
1. Randomly initialize \( G \) and \( D \)
2. for \( i = 1, 2, \ldots, K \) do
3. \( \) Sample a real datum \( x \) from \( X \)
4. \( \) Sample one random vector \( z \) from a uniform distribution
5. \( \) Train \( D \) to separate \( x \) and \( G(z) \)
6. \( \) Sample one random vector \( z \) from a uniform distribution
7. \( \) Train \( G \) to classify \( G(z) \) as a real datum
8. end for
```
### Table 2: Configuration of the GAN Model

| Name               | Type          | Shape          |
|--------------------|---------------|----------------|
| Generator input    | Dense         | 100            |
|                   | Reshape       | 6272           |
|                   | Batch normalization | 7×7×128  |
|                   | ReLU          |                |
|                   | ConvTransposed2D |             |
|                   | Batch normalization | 14×14×128 |
|                   | ReLU          |                |
|                   | ConvTransposed2D |             |
|                   | Batch normalization | 28×28×64 |
|                   | ReLU          |                |
|                   | ConvTransposed2D |             |
|                   | Batch normalization | 28×28×32 |
|                   | ReLU          |                |
|                   | ConvTransposed2D |             |
|                   | Batch normalization | 28×28×1  |
|                   | ReLU          |                |
| Generator output   |               | 28×28×1        |

| Discriminator input | LeakyReLU  | 28×28×1 |
|                     | Conv2D     | 14×14×32 |
|                     | LeakyReLU  | 7×7×64  |
|                     | Conv2D     | 4×4×128 |
|                     | LeakyReLU  | 4×4×256 |
|                     | Flatten    | 4096    |
|                     | Dense      | 4096    |
|                     | Sigmoid    | 1       |
| Discriminator output|           | 1       |

In this case, the training batch is set to 1 and only one fake datum is generated by $G$ when $G$ and $D$ are being trained. Lines 3 to 5 are the training procedures for $D$. In line 3, real data $x$ will be sampled from the training data. In line 4, a vector $z$ is sampled from a uniform distribution. In line 5 fake data is generated by the input of $z$ to $G$ denoted as $G(z)$, and $x$ is marked as true data and $G(z)$ as fake data, then the training of $D$ is started to separate $x$ and $G(z)$ as a binary classifier. Lines 6 and 7 are the $G$ training processes. In line 6, a vector $z$ is sampled from a uniform distribution. In line 7 the fake data $G(z)$ label is marked as real data, and $G$ is trained to try to deceive $D$. A powerful $G$ will mislead $D$ and recognize $G(z)$ as real data, and the prediction of $G(z)$ will match its label. Iteration of lines 3 to 7 will cause a powerful $G$ to be formed, and $D$ will be unable to tell the difference between $x$ and $G(z)$.

**D. The Proposed Training Framework**

The proposed chatter detection model needed to be tested for robustness and because the dataset was very small (only 60 data) it would have been unreasonable to separate it into training and testing datasets. Therefore, leave-one-out cross-validation was used. The training procedures used for $G$ are shown in Fig. 8. The testing data was not included in training the generator, because it would have influenced the validity of the test results. To generate the same amount of data for each class, it was necessary to train two separate generators. The one used to generate true data was denoted as $G_{Yes}$, and the other for the generation of false data was denoted as $G_{No}$. The 59 training data were separated into “Yes” and “No” data, and then $G_{Yes}$ and $G_{No}$ were trained independently. To run the leave-one-out cross validation, these procedures needed to be repeated sixty times to create sixty pairs of $G_{Yes}$ and $G_{No}$. FOCNN was trained after the two generators had been trained and a batch of random vectors were sampled from a customized uniform distribution and fed into $G_{Yes}$ and $G_{No}$ to generate the same amount of “Yes” and “No” data as shown in Fig. 9. These generated data were added to the training data to provide a big enough dataset to train FOCNN. Algorithm 2 describes the procedures of training FOCNN with GAN step-by-step. Lines 3 and line 4, produce equal amounts of true and false data. In line 5, three types of data are merged into $X_{train}$. Finally, FOCNN was trained by $X_{train}$ for $J$ epochs.
Fig. 8. Flowchart showing the splitting of data and the training generators.

Algorithm 2: Train FOCNN with GAN

Input: $X_o$ ← original training data;
      $G_{Yes}$ ← generator of true data
      $G_{No}$ ← generator of false data
      $gen$ ← the total amount of generated data
      $J$ ← training epochs

Output: FOCNN ;

1. Randomly initialize FOCNN
2. $Z$ ← sample $gen/2$ random vectors from a uniform distribution
3. $X_{Yes} = G_{Yes}(Z)$
4. $X_{No} = G_{No}(Z)$
5. $X_{train} = [X_{Yes}, X_{No}, X_o]$
6. for $i = 1, 2, ..., J$ do:
   7. Train FOCNN by $X_{train}$
8. end for

Fig. 9. The data generation process.
IV. THE EXPERIMENTAL RESULTS

In this section experimental results from the proposed method are compared with those from different models and the generation of augmented data is also discussed. The neural networks in this work were implemented in Keras with the TensorFlow backend.

To gain insight into the data generators, the data generated during the training processes are visualized in Fig. 10 and Fig. 11. In the experiments, the training iterations $K$ were set to 10,000 to preserve convergence. At the beginning of training, the generated data were rather vague. The shapes of visualized features were blurry and it was difficult to recognize features. By step 500, the data profiles were being gradually formed. Black pixels aggregated to a particular area. By step 1000, the profiles had become a lot clearer. After step 2500, the contours of features were almost fixed and clear which meant the generator model was converging. This series of images demonstrate that GAN can be used to train an eligible generator to produce data similar to original data.

![Generated "Yes" data at different steps.](image)

![Generated "No" data at different steps.](image)

![Accuracy of different numbers of generated data.](image)

The comparisons of the experimental results focused on two points: different ways to create data and different models. Data augmentation is a common way to create various types of data via rotation, scaling, shifting, whitening, etc. The computational
cost of data augmentation is far less than that of GAN. Therefore, data augmentation was taken into consideration to testify the need for GAN. The chatter dataset was also trained on an ordinary CNN and MLP to compare their feature extraction capability. Other common machine learning-based methods including SVM [42], Random forest (RF) [43], and Decision tree [44] were also compared with the proposed model. The parameters used with FOCNN, CNN, and MLP were 545, 1133, and 48061, respectively. The linear kernel and regularization parameter settings used for SVM were 0.5. The structure of CNN is almost the same as FOCNN except for the convolutional kernels. The maximum depth of RF used was 1000, and the number of trees was 100. The ReLU activation function was used in MLP, and the dropout rate was set to 0.2. Hyperparameters for the training of FOCNN, CNN and MLP were as follows: the maximum number of epochs was set to 30, the Stochastic Gradient Descent (SGD) was chosen as the optimizer, and the training batch was 1. There are only 60 data, so leave-one-out cross-validation was the reasonable approach to validate such a small dataset and the proposed method. Leave-one-out cross validation was carried out 10 times to provide a rigorous average accuracy.

TABLE 3 lists the average accuracy of 10 runs for different combinations of methods and models. The proposed method outstrips the other methods and FOCNN, MLP and CNN all had lower accuracy. Although MLP had the highest parameters, its feature extraction efficiency was lower than the CNN-based models. There was no doubt that CNN-based models had great feature extraction capacity due to the feature maps created by the convolutional layers. Feature maps can capture details of images by the use of different filters. Furthermore, the characteristics of the convolutional layer enable CNN to share parameters to give a lightweight model. FOCNN is a special case of CNN designed especially for our data and was able to extract more meaningful features than ordinary CNN. FOCNN not only offered the highest accuracy but also had the smallest model parameters. It was clear that FOCNN was a suitable model for a combination with GAN for data augmentation.

Different amounts of generated data (gen) were used in the experiments, and the results are shown in Fig. 12. When gen = 59, the two methods have the same validation accuracy and other gen numbers show different trends. The accuracy of GAN rose with an increase in the value of gen to a turning point when gen = 944, which was selected as it gave the best GAN average accuracy. On the other hand, it was found that the accuracy of data augmentation declined with an increase in gen. Although there was a slight increase in accuracy when gen = 1888, it was still far less than when gen = 0. The results showed that generating more data, by data augmentation, could not improve the performance and GAN was a better way to generate data than data augmentation.

To explain why data augmentation did not work on the chatter dataset, three types of data were projected in two-dimensional space using t-distributed stochastic neighbor embedding (t-SNE) as shown in Fig. 13 [45]. It can be seen that the original data and GAN data are scattered mostly to the left and right sides of the swarms and augmented data distributed in the middle of the plane. The augmented data deviated from the distribution of the original data and this was the main reason for the decline in accuracy. Training using a large proportion of unrelated data definitely influenced the result. Clearly, data augmentation was not the appropriate approach for the chatter dataset because arbitrary augmentation could easily distort the features of the data. GAN, on the other hand, aims to minimize Jensen-Shannon divergence (JS divergence) between distribution of the original data and that from the generator, which closely approximates the distribution of the original data. There are swarms of GAN data which are adjacent to (but not overlap) the original data. These were the key to increased accuracy and the generator could “guess” data that had not been collected. Although these data are not real, they are close enough to being real to improve the model. This step compensates for the sparsity of real data. GAN and data augmentation can both produce more data, but only GAN can generate data that is versatile enough and not too far from the original distribution.

The confusion matrix is a well-known metric to explain the testing results of classification and reflects the testing quality of a model. The confusion matrices of FOCNN, CNN, and FOCNN with GAN are shown in Fig. 14. The higher true positive and true negative values demonstrate that the model can correctly classify data with high reliability. Tricky data that can make a model misclassify can be handled with the help of additional data generated by GAN. This can be viewed as a breakthrough of FOCNN achieved without the need to modify a neural networks structure or collect more chatter data. Confusion matrices of the other common machine learning-based methods and MLP are shown in Fig. 15 (using the original one-dimensional raw data).

The results indicate that traditional machine learning-based methods are incapable of classifying complicated data correctly. Although SVM successfully classified all the false data, it performed very poorly on true data. Bias occurs in these models that cannot be avoided with these methods.
Fig. 13. Distribution of the different kinds of data projected by t-SNE.

Fig. 14. Confusion matrix—(a) FOCNN with GAN, (b) FOCNN, and (c) CNN.

Fig. 15. Confusion matrix—(a) MLP, (b) SVM, (c) Random Forest, and (d) Decision tree confusion matrix.
V. CONCLUSION AND FUTURE WORKS

In this study a new method was devised that improves the performance of FOCNN. The method provides a new framework for the improvement of FOCNN without a need for the collection of large amounts of data. The generator can produce versatile and high-quality data to fix the problem of the lack of data and directly improves the inference ability of FOCNN. Some difficult data which are hard to discriminate in a model can be correctly classified. The light structure makes real-time computation possible in an embedded system rather than a workstation. The trained FOCNN can be easily implemented in industry for chatter detection. There are several novel neural networks and GAN frameworks available for different purposes. In the future, experiments can be carried out with different kinds of models and GAN with other chatter datasets.

DATA AVAILABILITY

The datasets generated and analyzed during the current study are not publicly available due to the agreement with the government's project but are available from the corresponding author on reasonable request after the project completing.

FUNDING

This work was supported by the Ministry of Science and Technology, Taiwan, under Grants MOST 109-2221-E-194-050.

COMPLIANCE WITH ETHICAL STANDARDS

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

REFERENCES

1. Quintana, G., Curana, J.: Chatter in machining processes: A review. Int. J. Mach. Tools Manuf. 51, 363–376 (2011). https://doi.org/10.1016/j.ijmachtools.2011.01.001
2. Yan, Y., Xu, J., Wiercigroch, M.: Estimation and improvement of cutting safety. Nonlinear Dyn. 2019 984. 98, 2975–2988 (2019). https://doi.org/10.1007/S11071-019-04980-0
3. Yuan, L., Pan, Z., Ding, D., Sun, S., Li, W.: A Review on Chatter in Robotic Machining Process Regarding Both Regenerative and Mode Coupling Mechanism. IEEE/ASME Trans. Mechatronics. 23, 2240–2251 (2018). https://doi.org/10.1109/TMECH.2018.2864652
4. Hou, H., Yu, X., Xu, L., Chuai, R., Cao, Z.: Discrete-time terminal sliding-mode tracking control with alleviated chattering. IEEE/ASME Trans. Mechatronics. 24, 1808–1817 (2020). https://doi.org/10.1109/TMECH.2019.2928642
5. Lin, S., Zhang, W.: Chattering reduced sliding mode control for a class of chaotic systems. Nonlinear Dyn. 2018 934. 93, 2273–2282 (2018). https://doi.org/10.1007/S11071-018-4324-9
6. Li, D., Cao, H., Liu, J., Zhang, X., Chen, X.: Milling chatter control based on asymmetric stiffness. Int. J. Mach. Tools Manuf. 147, 103458 (2019). https://doi.org/10.1016/j.ijmachtools.2019.103458
7. Totis, G., Sortino, M.: Polynomial Chaos-Kriging approaches for an efficient probabilistic chatter prediction in milling. Int. J. Mach. Tools Manuf. 157, 103610 (2020). https://doi.org/10.1016/j.ijmachtools.2020.103610
8. Nam, S., Eren, B., Hayasaka, T., Sencer, B., Shimoto, E.: Analytical prediction of chatter stability for modulated turning. Int. J. Mach. Tools Manuf. 165, 103379 (2021). https://doi.org/10.1016/j.ijmachtools.2021.103379
9. Aslan, D., Altintas, Y.: On-line chatter detection in milling using drive motor current commands extracted from CNC. Int. J. Mach. Tools Manuf. 132, 64–80 (2018). https://doi.org/10.1016/j.ijmachtools.2018.04.007
10. Yesilli, M.C., Khasawneh, F.A., Otto, A.: On transfer learning for chatter detection in turning using wavelet packet transform and ensemble empirical mode decomposition. CIRP J. Manuf. Sci. Technol. 28, 118–135 (2020). https://doi.org/10.1016/j.cirp.2019.11.003
11. Sun, Y., Xiong, Z.: An optimal weighted wavelet packet entropy method with application to real-time chatter detection. IEEE/ASME Trans. Mechatronics. 21, 2004–2014 (2016). https://doi.org/10.1109/TMECH.2016.2547481
12. Thaler, T., Potočnik, P., Bric, I., Govekar, E.: Chatter detection in band sawing based on discriminant analysis of sound features. Appl. Acoust. 77, 114–121 (2014). https://doi.org/10.1016/j.apacoust.2012.12.004
13. Nguyen, V., Melkote, S., Deshamadre, A., Khanna, M., Walker, D.: Comparison of on-line chatter detection algorithms in turning. Int. Symp. Flex. Autom. ISFA 2016. 87–94 (2016). https://doi.org/10.1109/ISFA.2016.7790141
14. Ding, L., Sun, Y., Xiong, Z.: Early chatter detection based on logistic regression with time and frequency domain features. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, AIM. 1052–1057 (2017). https://doi.org/10.1109/AIM.2017.8014158
15. Wan, S., Li, X., Yin, Y., Hong, J.: Milling chatter detection by multi-feature fusion and Adaboost-SVM. Mech. Syst. Signal Process. 156, 107671 (2021). https://doi.org/10.1016/j.ymssp.2021.107671
16. Shao, Q., Feng, C.J.: Pattern recognition of chatter generation based on Hybrid PCA-SVM. Appl. Mech. Mater. 120, 190–194 (2012). https://doi.org/10.4028/www.scientific.net/AMM.120.190
17. Yao, Z., Mei, D., Chen, Z.: On-line chatter detection and identification based on wavelet and support vector machine. J. Mater. Process. Technol. 210, 713–719 (2010). https://doi.org/10.1016/j.jmatprotec.2009.11.007
18. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nat. 2015 5217553. 521, 436–444 (2015). https://doi.org/10.1038/nature14539
19. Pak, M., Kim, S.: A review of deep learning in image recognition. Proc. 2017 4th Int. Conf. Comput. Appl. Inf. Process. Technol. CAIPT 2017. 2018–January, 1–3 (2018). https://doi.org/10.1109/CAIPT.2017.8320684
20. Guo, Y., Liu, Y., Orefolans, A., Lao, S., Wu, S., Lew, M.S.: Deep learning for visual understanding: A review. Neurocomputing. 187, 27–48 (2016). https://doi.org/10.1016/j.neucom.2015.09.116
21. Nassif, A.B., Shahin, I., Attili, I., Azzeb, M., Shaalan, K.: Speech Recognition Using Deep Neural Networks: A Systematic Review. IEEE Access. 7, 19143–19165 (2019). https://doi.org/10.1109/ACCESS.2019.2896880
22. Sezer, B., Serin, G., Guðdekk, M.U., Murat Özayoglu, A., Unver, H.O.: Intelligent Chatter Detection in Milling using Vibration Data Features and Deep Multi-Layer Perceptron. Proc. - 2020 IEEE Int. Conf. Big Data, Big Data 2020. 4759–4768 (2020). https://doi.org/10.1109/BigData50022.2020.9378223
23. Arena, P., Calf, M., Patané, L., Portera, A., Spinosa, A.G.: A CNN-based neuromorphic model for classification and decision control. Nonlinear Dyn. 2018 953. 95, 1999–2017 (2018). https://doi.org/10.1007/S11071-018-4673-4
24. Landauskas, M., Cao, M., Ragulskis, M.: Permutation entropy-based 2D feature extraction for bearing fault diagnosis. Nonlinear Dyn. 2020 1023. 102, 1717–1732 (2020). https://doi.org/10.1007/S11071-020-06604-6

25. Tran, M.-Q., Liu, M.-K., Tran, Q.-V.: Milling chatter detection using scalogram and deep convolutional neural network. Int. J. Adv. Manuf. Technol. 2020 1073. 107, 1505–1516 (2020). https://doi.org/10.1007/S00170-019-04807-7

26. Sener, B., Gudelek, M.U., Ozayoglu, A.M., Unver, H.O.: A novel chatter detection method for milling using deep convolution neural networks. Measurement. 182, 109689 (2021). https://doi.org/10.1016/J.MEASUREMENT.2021.109689

27. Vashisht, R.K., Peng, Q.: Online Chatter Detection for Milling Operations Using LSTM Neural Networks Assisted by Motor Current Signals of Ball Screw Drives. J. Manuf. Sci. Eng. 143, (2021). https://doi.org/10.1115/1.4048001

28. Shi, F., Cao, H., Wang, Y., Feng, B., Ding, T.: Chatter detection in high-speed milling processes based on ON-LSTM and PBT. Int. J. Adv. Manuf. Technol. 2020 11111. 111, 3361–3378 (2020). https://doi.org/10.1007/S00170-020-06292-9

29. Altintas, Y., Stepan, G., Budak, E., Schmitz, T., Kilic, Z.M.: Chatter Stability of Machining Operations. J. Manuf. Sci. Eng. 142, (2020). https://doi.org/10.1115/1.4047391

30. Jian, B.-L., Wang, C.-C., Chang, J.-Y., Su, X.-Y., Yau, H.-T.: Machine Tool Chatter Identification Based on Dynamic Errors of Different Self-Synchronized Chaotic Systems of Various Fractional Orders. IEEE Access. 7, 67278–67286 (2019). https://doi.org/10.1109/ACCESS.2019.2917094

31. Chen, H.K., Lee, C.I.: Anti-control of chaos in rigid body motion. Chaos, Solitons and Fractals. 21, 957–965 (2004). https://doi.org/10.1016/J.CAMWA.2011.03.054

32. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-December, 770–778 (2015)

33. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve restricted boltzmann machines. In: Icml (2013)