Vibrational Spectroscopy 121 (2022) 103391

Contents lists available at ScienceDirect

Vibrational Spectroscopy

journal homepage: www.elsevier.com/locate/vibspec

# Single convolutional neural network model for multiple preprocessing of Raman spectra

Jiahao Shen, Miao Li, Zhongfeng Li, Zhuoyong Zhang, Xin Zhang *

*Department of Chemistry, Capital Normal University, Beijing 100048, China*

ARTICLE INFO

*Keywords:*
Convolutional Neural Network (CNN)
Raman spectroscopy
Spectral preprocessing
Deep Learning
Multiple data preprocessing

ABSTRACT

Raman spectra suffer from the interferences of noises, baseline drifts, and cosmic rays, which leads to errors in the subsequent analysis of the spectra. The commonly applied spectral preprocessing methods, such as wavelet transform (WT), Savitzky-Golay smooth (SG smooth), asymmetric least squares (AsLS) can only correspondingly reduce a single interference term, and the completion of preprocessing based on these traditional methods requires a series of tedious trials. Especially, each scheme can only be used for a specific data set. Convolutional neural network (CNN) is commonly used for object recognition, image super-resolution, and natural language processing. CNN has the potential to be applied to the preprocessing of chemical signals to solve problems intelligently. In this paper, we developed a method to remove all the interferences from multiple Raman spectra data sets collected from food and wastewater samples, using only a single CNN model. By optimization of hyperparameters, activity functions, and loss functions in CNN, the CNN model completely removed noises, spikes, baselines, and cosmic rays from Raman spectra and simplified the preprocessing of Raman spectra.

## 1. Introduction

Raman spectroscopy is a rapid and non-destructive analytical technique, provides fingerprint information of samples in complex systems by analyzing their molecular information. It has attracted considerable attention in different fields, such as food, medicine, and the environment [1–4]. However, Raman spectra suffer from the interferences from noises, and cosmic rays which are caused by the environment, sensors, laser irradiation, and random cosmic ray during the measurement. Chemometrics models are needed to extract the quantitative and qualitative information from Raman spectra. In order to enhance spectral features of interest, the preprocessing methods have been developed to denoise, remove baselines, and correct spikes [5], like Savitzky-Golay smooth (SG smooth) [6], asymmetric least squares (AsLS) [7], wavelet transform (WT) [8,9], moving average smoothing (MAS) [10], and empirical mode decomposition (EMD) [11]. These methods are based on polynomial fitting [12,13], penalized least square [14,15], peak detection and interpolation methods [16], wavelet transform [17] and morphological operations [18], etc. The classical spectral preprocessing methods mentioned above do well in the pretreatment of Raman spectra in eliminating noises, baseline drifts, and spikes. But a single method can only process one corresponding interference item, and a series of methods can only correct a specific data set of the training samples with their optimized parameters. For example, SG smooth is a prevalent preprocessing method based on local least squares polynomial approximation in a window for denoising of spectra. When applying SG smooth, the order of the polynomial and the size of the smoothing window are critical for noise reduction. AsLS is a commonly used method to estimate the baseline based on Whittaker smoother [19], while the optimization of the parameters of AsLS is very tricky and difficult to be obtained without experience in this method. Most other methods for spectra preprocessing also need to optimize parameters for different data sets. To improve the spectra collected from different samples with diverse characteristics and interferences, some newly optimized parameters are needed for the preprocessing of each data set. Renewing the parameters is a very cumbersome process, while it is still difficult to obtain a well processed spectrum when confronting complex background interferences. It is interesting to develop a method with the potential to correct all the interferences in only one step.

Deep learning is an extremely hot research direction in machine learning in the past few years [20–23]. With rapid development, deep learning has been widely used in different fields such as natural language processing [24], and computer vision [25] for its strong feature learning ability. Recently, deep learning approaches have been

* Corresponding author.
*E-mail addresses:* xinzhang@cnu.edu.cn, xinkevinzhang@outlook.com (X. Zhang).

https://doi.org/10.1016/j.vibspec.2022.103391
Received 3 December 2021; Received in revised form 24 May 2022; Accepted 5 June 2022
Available online 10 June 2022

employed to tackle the issue of background removal in spectra [26]. Before the popularity of deep learning, artificial neural network (ANN) has achieved considerable satisfactory prediction for classification and regression in spectral analysis. ANN was inspired by biological neural response modes [27]. The neural network models based on multi-layer perceptron (MLP) were built at an early stage, and back-propagation artificial neural network (BP-ANN) was developed later. With the progress of the theory and the improvement of computing power, convolutional neural network (CNN) was developed. Many neural network models based on CNN have won some important competitions in machine learning and have attracted great attention to researchers all around the world [28,29]. CNN is a feedforward neural network with a convolution algorithm and is now widely used in deep learning methods [22,30–32]. The excellent performance of CNN benefits from the special network structure with convolution and pooling layers, which enables its potential to well extract and learn the characteristics of spectral data and achieve high accuracy in subsequent qualitative and quantitative models [31,33]. In addition, CNN can effectively reduce training weights and error attenuation and speed up the calculation, through local connections and weight sharing, which help to avoid the defects of multilayer neural networks.

In recent three years, Deep CNN has been used for the preprocessing of Raman spectra and the classification of near-infrared (NIR) spectra and Raman spectra [22,31,32,34]. Such methods were also compared with logistic regression (LR), k-nearest neighbor (KNN), random forest (RF), and back-propagation artificial neural network (BP-ANN) models. CNN has also been applied for the variable selection for the modeling based on spectra. Important regions of the spectrum were selected based on the weights in the pooling layer and the fully connected layer [31]. High weight values were assigned to the neural carrying common characteristic of Raman spectra collected from different samples. Joel Wahl et al. studied the application of convolutional neural networks in the preprocessing of Raman spectra [22]. Cosmic rays, noises, baseline drifts can be decreased in a single preprocessing step by CNN. However, a method that can correct the spectra collected from different substances or complex systems in nature is still interesting to improve the analysis of mixtures of different compounds.

In this paper, we present a new method to reduce all the mentioned interferences of multiple Raman data sets in only one step by using the CNN. The proposed method can also cope with the difficult situation that Raman spectra have different numbers of variables and various samples collected by different instruments. The features of the spectra extracted by the hidden layers were discussed. Our CNN simplified the preprocessing step of Raman spectra, reduces the processing time, and provided a new idea for preprocessing method of Raman spectra.

## 2. Materials and methods

### 2.1. Materials

The first Raman spectra data was collected from compressed milk tablet candy with calcium, vitamin A and D additives. Fifteen spectra were collected with 2075 variables ranging from 250 to 2339 $cm^{-1}$. This data set has been used for secured signal transmission by time-variant sensing matrices [35].

The second data set [36] is Raman spectra of water samples gathered from hospitals and wastewater treatment plants in Wuhan (China) from 24th March to 10th April 2020. Angiotensin-converting enzyme 2 @Silver-nanorod surface-enhanced Raman scattering (ACE2 @SN-SERS) was developed to detect the existence of SARS-CoV-2 in environmental samples. ACE2 @SN-SERS substrate can generate strong Raman signals. If SARS-CoV-2 spike protein is captured by ACE2, the Raman signal would be quenched by either red-shift or whole spectral alterations. This can be used as a method to detect SARS-CoV-2. The seventeenth water samples were acquired by a NIR confocal Raman microscope (HR evolution, Horiba, USA) equipped with a 785 nm NIR laser source, a 300 l/mm grating, and a semiconductor-cooling detector (CCD). All Raman spectra were acquired with 50 times objective lens (NA = 0.7), 10 s exposure time, and 3 accumulations. Laser power was 10 mW. Before spectra acquisition, Raman spectroscopic system was calibrated using a silicon wafer at Raman shift of 520 $cm^{-1}$. The raw spectra data set was cut into a biochemical-cell fingerprint region (900–1800 $cm^{-1}$, 900 variables) and preprocessed by baseline correction, wavelet de-noising, and vector normalization.

The third Raman spectra data set was collected from tomatoes to classify frostbitten and intact tomatoes. The instrument used in this experiment was a Renishaw inVia Raman spectrometer with a 785 nm laser source, 40 mV laser power. The exposure time was 1 s. The frostbitten tomatoes data set used in this research has 50 spectra with 1015 variables (range in 505.716–1631.18 $cm^{-1}$).

### 2.2. Data simulation

Considering the Raman spectra datasets were collected from areas in food, and environment suffering different interferences, while a clean spectra dataset would be helpful for the evaluation of the CNN model, the spectra from the three datasets were first preprocessed by traditional methods like S-G smooth, AsLS and de-spike method. This clean data can be used for the evaluation of the output of the CNN model optimized. In the next steps, we added random noise, baselines, and spikes at a high level to train the CNN, and the interference was well designed, which could make the performance of CNN under control. To expand the volume of the dataset padding were used. Each spectrum was moved left or right a few wavenumbers randomly. The spectra with a different number of variates were arranged in a matrix, so that all of them can be input into a unique model. The data simulation or augmentation methods is specially designed in considering processing the unknown samples collected from instruments with different resolutions.

The scheme for generating data is summarized in the following four steps:

(1) Preparation of the target data: The three data sets are normalized and preprocessed by AsLS and SG smooth to remove baseline drift and noises. No cosmic ray exit in the raw data sets. These three preprocessed data sets could be considered as target data sets of the CNN. The interferences were added to this cleaned data for the training of CNN.

(2) Adding baseline drift to each target data set:

$$y = ax^2 + b \tag{1}$$

$$\text{data\_b} = \quad data \quad + \quad y \tag{2}$$

where Eq. 1 is a quadratic function, $x$ is the sequence number of the variables, $y$ is the simulated baseline, $a$ is a random slope to control the level of the baselines, and $b$ is a random intercept to set the location of the profiles. The Equations illustrate the way of baseline drift simulation. *data* in Eq. 2 is a target data set and *data_b* is the simulated data set affected by baseline drift.

(3) Simulation of a complex system and expanding the size of the data set: Each *data_b* matrix was stacked with its mirror flipped matrix to simulate more complicated cases with unknown interferences or a variety of baseline drifts in the analysis of complex systems. Then, we added Gaussian noises to the stacked data. The standard deviation of Gaussian noises is 1, the mean value is 0 and the maximum value of Gaussian noises is 2% of the maximum value of the target data. The *data_b* influenced by Gaussian noises is named as *data_bn*.

(4) Simulation of spikes: Spike (or cosmic ray) is a random influential factor in Raman spectra. Commonly it must be removed before any other preprocessing and modeling of Raman spectra. Not every spectrum is disturbed by spikes, thus, it is difficult to extract the characteristic of spikes by CNN. Spikes were added at random locations on a random spectrum of a data set affected by noises. The intensity of the simulated spike intensity is between one and two times the maximum of the

standardized spectra. The simulated *data_bn* influenced by spikes is *data_bns*.

In practice, when the Raman spectra are measured in a way of imaging on complex samples, or analyzed by the spectroscopy instruments of different brands produced by factories, the spectra have different numbers of variables for the compounds and suffer from different levels of interferences. When complex data sets are needed to be pretreated, a powerful one-step method would be useful. We firstly arranged Raman spectra with different numbers of variables before training the neural network.

In order to make data matrices of the three data sets, compressed milk tablet candy, SARS-CoV-2, and tomatoes data sets, have the same number of variables in wavenumber direction, we investigated two ways to align the Raman spectra to a big matrix with 4000 variables, which is sufficiently long to spectra collected by most of the instruments: (1) we expanded the number of the variables by duplicating the spectrum having fewer wavenumbers than the target matrix variables and copying the spectrum to its end till the target variable number; (2) we added variates near to zeros (or numbers in noise level) to the end of Raman spectra whose number of variables is less than others. Different from zero padding in convolution operation, zeros added to the spectra in our method were aligned the spectra to the same length to make data input and training convenient. The simulated spectra are shown in θ. 1.

Ten thousand spectra were simulated based on the Raman spectra collected from compressed milk tablet candy, SARS-CoV-2, and tomatoes samples, respectively. These three simulated data sets were all used for unique CNN model training. After data simulation, each simulated data set contains 10,000 simulated Raman spectra, in sum, 30,000 simulated Raman spectra were used as the input for CNN training. 75% of the three datasets (22,500 spectra) were taken as the training set. The remaining 25% (7500 spectra) in three datasets were used as the external test set. One out of every 15 spectra (1500 spectra) in the training set were used as the validation set for the optimization of hyperparameters of CNN. Kennard-Stone method was used for subset partitioning.

### 2.3. Method

In this section, we describe the proposed CNN based deep learning model. The simulated noisy spectra collected from different dataset are saved in a unique matrix and used as the input of the CNN. CNN produce a hierarchy of latent feature maps via learned filters. The spectra feature can be recognized and separated from the noise, baselines and spikes by optimizing the parameters in CNN under the supervision of the prepared clean spectra. The output of CNN is still with similar size of the input, and the processed spectra with similar number variates in wavelength of the original spectra will be cut out from the corresponding elements of the output matrix.

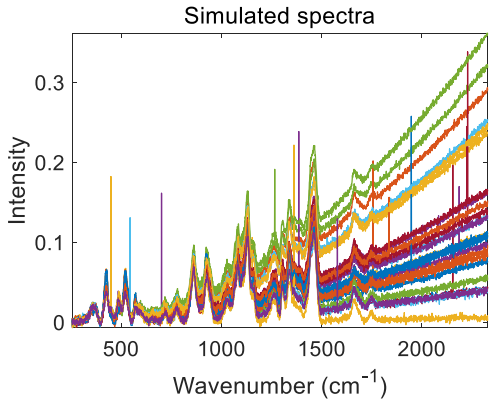


**Fig. 1.** Simulated dataset to expand the dataset have similar numbers of variables.

Fig. 2 shows schematics of the preparation of the dataset and the optimized neural network. When we train the CNN, we optimized the number of layers, the kind of activate function, loss function, and optimizer by using one dataset. Then, the filter size, the number of channels, and standard deviation were optimized by using a mixed dataset.

The optimized CNN model was built with four convolution blocks and three transposed convolution blocks. The first three convolution blocks have a convolution layer (Conv1 to Conv3 in Table 1) applied convolution to its input, following an activation layer, a dropout layer, and a Maxpooling layer. The last convolution block uses a convolution layer (Conv4), an activation layer. The last convolution block has an activation layer and a dropout layer. The algorithm of convolution layers that is responsible for feature extraction can be explained by Eq. 3.

$$X_j^l = f\left(\sum_{i\in M_j} X_i^{l-1} \otimes K_{ij}^l + b_j^l\right) \tag{3}$$

where $X_i^{l-1}$ is the output of $l-1$ layer, $K_{ij}^l$ is the kernel whose size is i × j in layer $l$, and $b_j^l$ is the bias in layer $l$. $M_j$ is all the input data. $X_j^l$ is the output of layer $l$. In the convolution process, the convolution filter moves sequentially across the input spectrum, meanwhile, the features in the $X_i^{l-1}$ are weighted. Important features in the spectra would get high weight values.

ReLU, a commonly used activation function in artificial neural networks, is used in activation layers to increase the non-linearity of the convolution layers. It redefined the output (X_i) of the previous layer by setting all negative values to zero and remaining the positive values (Eq. 4).

$$\widehat{X}_i = max(0, X_i) \tag{4}$$

$X_i$ is the output of the input and $\widehat{X}_i$ is the output of the activation layer. This operation makes CNN no longer a linearly stacked neural network, but a network that can solve a non-linear problem.

Max-pooling layers are used to retain high-weight features and discard low-weight features to reduce the amount of data. The size of the max-pooling kernel is 2 × 2. In the maxpooling layer, the 2 × 2 moving window moves sequentially across the input, keeping only one maximum value in the window. After one maxpooling layer, the size of the input is changed to one-half of what it was before.

The dropout method is applied in our CNN. This method randomly set 30% of the neurons in the current layer to zero. The purpose of the dropout method is to prevent overfitting and reduce the number of parameters in each training epoch.

After up sampling transposed convolution blocks, the variable of input corresponding to the features of each spectrum in the corresponding layer was reduced to 500 × 64. Different from regression model or classification model, our CNN was constructed to handle Raman spectra preprocessing, so the output from the CNN should have the same size to the input (the final spectra with longer variable of three data sets were cut). In order to gain a preprocessed spectrum with a similar size to the input, an upsampling method should be used here. Transposed convolution layers were applied in CNN to recover the features of the spectra. Transposed convolution block has three transposed convolution layers and each convolution layer follows an activation layer. The transposed convolution layer recovers the number of features extracted through the convolution and pooling layers to the resolution of the original feature. It can be used to perform upsampling at the same time as the training process, and to minimize the loss resulting from feature resampling.

Transposed convolution layers played a crucial role in up sampling using learnable parameters. It is a process that can be considered as an opposite process to any simple CNN and can be considered as the operation that allows to recover the shape of this initial feature map. So

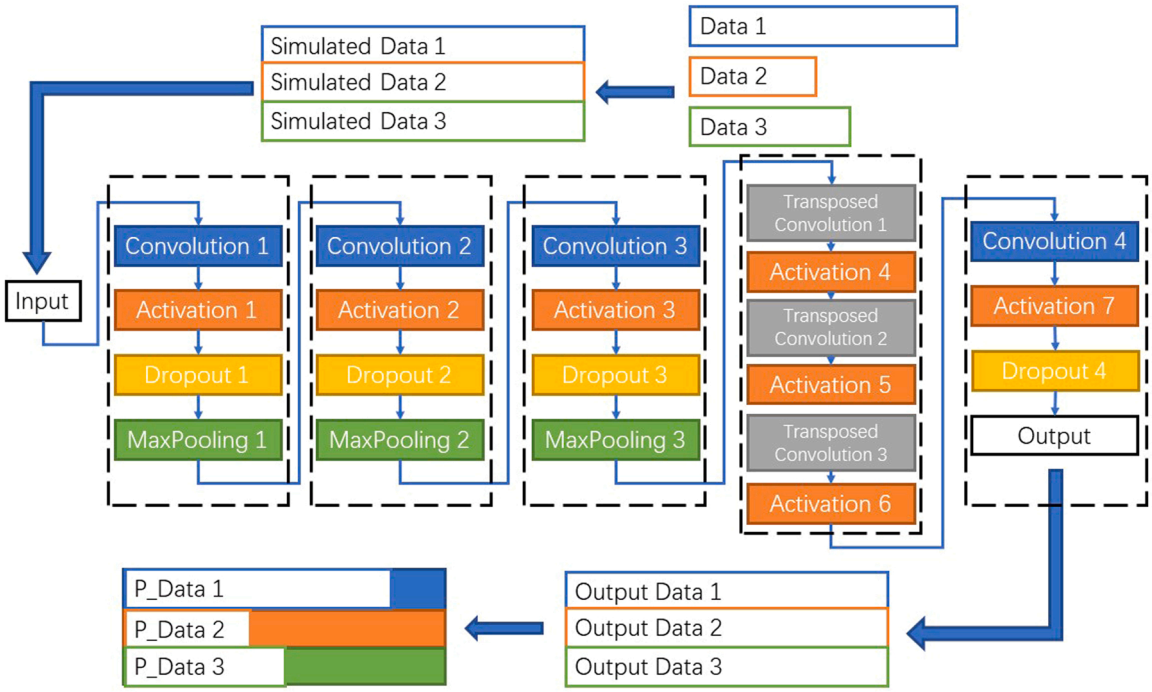


**Fig. 2.** Schematics of (from left to right) of the optimized deep learning model based on CNN.

**Table 1**
Parameters of the optimized deep learning model based on CNN applied for the preprocessing of Raman spectra.

| | Filter size | Channel | Strides | Standard deviation | Padding | Activation function | Dropout rate |
|---|---|---|---|---|---|---|---|
| Conv1 | 1 × 11 | 64 | 1 | 0.15 | SAME | ReLU | 0.3 |
| Conv2 | 1 × 9 | 64 | 1 | 0.15 | SAME | ReLU | 0.3 |
| Conv3 | 1 × 7 | 64 | 1 | 0.15 | SAME | ReLU | 0.3 |
| Deconv1 | 3 × 3 | 64 | 2 | 0.15 | SAME | ReLU | 0.3 |
| Deconv2 | 3 × 3 | 64 | 2 | 0.15 | SAME | ReLU | 0.3 |
| Deconv3 | 3 × 3 | 64 | 2 | 0.15 | SAME | ReLU | 0.3 |
| Conv4 | 1 × 3 | 1 | 1 | 0.15 | SAME | ReLU | 0.3 |
| MaxPooling | 2 × 2 | \ | 2 | \ | SAME | ReLU | 0.3 |

the CNN with transposed convolution layers returns a feature map that has the same width and height to the input. The filters of transposed convolution layers can be trained to learn the features of samples as filters of convolution layers did. The transposed convolution operation can be thought of as the gradient of some convolution with respect to its input, which is usually how transposed convolutions are implemented in practice.

The principle of transposed convolution is similar to convolution. The convolution filter moves sequentially across the transferred vectors from previous layers, meanwhile, the features in the corresponding layers are weighted. The only difference is that the weight in kernels of transposed convolution layers is different from that of convolution layers. The output shape of the transposed convolution layer can be controlled by setting the parameter named output shape. The more detail principles of transposed convolution can be found in [37]. The aim of transposed convolution layers in this work is to recover the size of the vector back to the size of input (1 × 4000).

In summary, the optimized CNN is comprised of four convolution layers (Conv), seven activation layers, three max-pooling layers (MaxPool), and three transposed convolution layers. Rectified Linear Unit (ReLU) is used for activation. The CNN model maps the interfered spectra to the target spectra (output). In our experiments, the input is simulated spectra (the input vector size is 4000 × 1). The learning rate and the epoch are 0.0005 and 30, respectively. Fifteen-fold cross validation was performed to train the CNN models. The hyperparameters of hidden layers are shown in Table 1. The inputs of the network are simulated spectra affected by, noises, baseline drifts, and cosmic rays. The outputs are preprocessed spectra. In real applications, experimental spectra will take the place of simulated data, and the preprocessed spectra will be outputted. To guarantee the satisfactory performance of the CNN model, the simulated data was shuffled into a random sequence. Then, the training sets were fed into the training network from input layers.

The loss function measures the errors when different hyperparameters are used in CNN learning. We compared the performance of using *L1_loss* and *L2_loss* as loss functions in CNN (Eqs. 5, 6).

$$L1\_loss = \frac{1}{R}\sum_{i=1}^{R}|Y_i - \widehat{Y}_i| \tag{5}$$

$$L2\_loss = \frac{1}{R}\sum_{i=1}^{R}(Y_i - \widehat{Y}_i)^2 \tag{6}$$

where $\widehat{Y}_i$ is the output of CNN, $Y_i$, is the target spectrum, and $R$ is the number of output spectra. MSE is always positive and it is close to zero when the outputs are near to the input spectra.

The optimizer is to update and calculate the hyperparameters that affect model training and model output, and they approximate or reach the optimal value by minimizing the loss function. Adam optimizer, RMSProp optimizer, gradient descent optimizer, and Adagrad optimizer

were compared in our CNN. These four optimizers are usually used in deep networks with different characteristics [38–40]. Matrix cosine similarity (MCS) is applied in this work to evaluate the similarity between the input and output spectra. As shown in Eq. 7, MCS is a simple and efficient method to evaluate the similarity between two matrices. A well-performing CNN gives a high similarity between the outputs and target spectra and would make the MCS close to one.

$$cos(\theta) = \frac{Y_i \cdot \widehat{Y}_i}{\|Y_i\|\|\widehat{Y}_i\|} = \frac{\sum_{i=1}^{n} Y_i \cdot \widehat{Y}_i}{\sqrt{\sum_{i=1}^{n} (Y_i)^2} \times \sqrt{\sum_{i=1}^{n} (\widehat{Y}_i)^2}} \tag{7}$$

Signal to noise ratio (SNR) and root mean square error (RMSE) can evaluate the performance of CNN models in our work. RMSE measures the similarity of the output data from the trained CNN and the target data. SNR measures the signal power over noise power (Eq. 8).

$$SNR = \frac{\sum_{i=1}^{R} Y_i}{\sum_{i=1}^{R} (Y_i - \widehat{Y}_i)^2} \tag{8}$$

RMSE is defined as Eq. 9, which gives the spread of the residuals of the output spectra:

$$RMSE = \frac{1}{R} \sqrt{\sum_{i=1}^{R} (Y_i - \widehat{Y}_i)^2} \tag{9}$$

where the $Y_i$ is the output and $\widehat{Y}_i$ is the target spectra, and R is the total number of output spectra.

Structural Similarity (SSIM) compares the similarity between two images [41,42] and here we apply this parameter for the comparison of two matrices, in order to evaluate the similarity between the output of the CNN and the target spectra. SSIM is defined as Eq. 10.

$$\text{SSIM} = \left(\frac{2\mu_{\widehat{Y}_i}\mu_{Y_i}}{\mu^2_{\widehat{Y}_i} + \mu^2_{Y_i}}\right)\left(\frac{2\sigma_{\widehat{Y}_i}\sigma_{Y_i}}{\sigma^2_{\widehat{Y}_i} + \sigma^2_{Y_i}}\right)\left(\frac{\text{cov}(\widehat{Y}_i, Y_i)}{\sigma^2_{\widehat{Y}_i} + \sigma^2_{Y_i}}\right) \tag{10}$$

where $\mu_{\widehat{Y}_i}$ is the average of the target spectra, $\mu_{Y_i}$ is the average of output data, $\sigma_{Y_i}$ is the standard deviation of target data, $\sigma_{\widehat{Y}_i}$ is the standard deviation of output data, $\text{cov}(\widehat{Y}_i, Y_i)$ is the covariance of output data and target data.

In this study, CNN was constructed on Python 3.8 using TensorFlow 2.2.0 with GPU acceleration. The hardware platform for the workstation is a Windows 10 laptop with Intel(R) Core (TM) i5–9300 H CPU, NVIDIA GeForce GTX 1660 Ti GPU, 16 GB RAM, and 1 TB SSD.

## 3. Results and discussions

### 3.1. Results for the models built for single data set respectively

The preprocessing CNN model was built using the single compressed milk tablet candy, SARS-CoV-2, and tomatoes data sets separately, to investigate the effects of hyperparameters and prevent interaction between the data sets. In this section, we investigated the hyperparameters, including the number of layers, the size of the filters, the loss functions, the optimizer, the change of the kernel, and the number of channels in the CNN.

**Figure SI 1** shows the partial examples of simulated data, the target data, and the output profiles of the CNN on the processing of compressed milk tablet candy data set. The CNN models built with different numbers of convolution layers were compared firstly. The results of the CNN were compared with the target data in **Figure SI 1**, and **Table SI 1**. Different levels of baseline drifts and random spikes were added to the target spectra to simulate the true situations when Raman spectra were collected. The CNN can easily remove different levels of baselines and random spikes simultaneously when processing a single data set. But, when the number of the convolutional layers of the model decreased to 1, the model lost the advantages of deep learning and cannot remove noises well. Using more convolution layers in the CNN model can make the output spectra smoother.

The number of layers was investigated to optimize the CNN models for spectra preprocessing. When 2, 3, and 4 CNN layers were used in the model, more layers in the CNN models can improve the smoothing performance of the neural network. However, the five-layer CNN can only give a mass of noise hardly provide any features of the spectra. The outputs of the five-layer CNN also cannot decrease baseline drifts, nor spikes. The four-layer CNN model achieved the best performance in smoothing, baseline drifts, and removing spikes. The two and three convolution layers models cannot capture sufficient spectral information details, while the five-layer model may face a risk of overfitting and cannot give satisfying correction in the test set. In each CNN block, a pooling layer was added to the model to limit the scale of the parameters needed to be optimized, while it may also lose more information while increasing the number of convolution layers.

Filter sizes of the receptive field of the convolution layers are essential for the feature capture in CNN. The larger filters can extract more spectral information than small filters, while the small filters can provide more detailed information on the spectra. When the filter sizes were $1 \times 11$, $1 \times 9$ and $1 \times 7$, $1 \times 3$, CNN gave a good performance in removing the noises, while the outputs from CNN with $1 \times 3$ filters were still influenced by noises. In our situation, the $1 \times 9$ filter extracted the details from the features of the $1 \times 11$ filter, the $1 \times 7$ filter extracted the details from the second convolution layer, and finally the $1 \times 3$ filter removed the low-level noises and generated the output data. Weight sharing is one of the characteristics of CNN and it can accelerate CNN training, meanwhile, gradient explosion can be prevented. The convolution kernel size directly determines the characteristic of feature maps the CNN extracted. The kernels are commonly square matrices when CNN is applied to the detection of images. In the computer vision area, the analyzing objects are images include three channels with space features, but in this work the objects are spectra with multiple variates. In this paper, we investigate the filter sizes of the CNN. As shown in **Figure SI 2** and **Table SI 2**, the gradually shrinking filters were used for the training of CNN models. One single spectrum was input into the CNN model in each training. A series of row vectors were used as filters in convolution layers and they can move across the spectrum tightly without introducing unexpected noises. When square metrics were used as filters in the CNN, more parameters are optimized using a high-cost calculation source, and extra noises may be induced.

In our experiments, *L1_loss* function and *L2_loss* function were compared to be applied as the loss function of CNN. The performance of Adam optimizer, RMSProp optimizer, gradient descent optimizer, and Adagrad optimizer were compared in the spectra preprocessing when *L1* and *L2* loss functions were used, respectively. The results obtained with different optimizers and loss functions are shown in **Figure SI 3** and **Table SI 3**. *L1_loss* or *L2_loss* provided almost similar results to all the models investigated. The optimizers performed differently in optimizing the models. The model using Adam optimizer with *L1_loss* or *L2_loss* achieved satisfied processed spectra. The results of the model applied RMSProp optimizer with *L1_loss* and *L2_loss* is also acceptable. Adam optimizer is based on a variant of the gradient descent algorithms. The learning rate of Adam optimizer is restricted to a certain range so that the value of the weight parameter is relatively stable without great fluctuation when a large gradient is encountered [38].

To make sure the model can process the spectra with different numbers of variables collected with instruments with different resolutions, we trained the CNN model by using vectors that have much more elements than the spectra of the three datasets. The empty elements were fulfilled in two ways, filling in zeros and repeating the processed spectra for training or testing (shown in the method section). The output

spectra of the two ways of aligning the spectra are shown in figure SI 4. The outputs of CNN trained using data aligned by repeating variables are shown in figure SI 4 (a1-a4). The outputs of CNN trained using data aligned by adding zeros in the wavenumber direction are shown in figure SI 4 (b1-b4). SNR, SSIM, RMSE, and MCS were used to evaluate the output data set based on two methods of aligning the spectra matrices. According to **Figure SI 4** and **Table SI 4**, both filling zeros and repeating variables for CNN training can make the neural network process the Raman spectra with different numbers of variables. All the output spectra have been removed baseline drifts, noises, and spikes, and remaining the characteristic peaks.

The parameter of the level of standard deviation was used to control the variation range of the convolution kernel in the convolution layer during the training. Setting a large standard deviation can increase the degree of fluctuation of the values in the convolution kernel and may give greater randomness to the output spectra. If the weight changes of the convolution kernels are too small, the CNN model will be difficult to achieve the goal within a limited training time. In our experiments, the standard deviation setting of 0.15 gave outputs with high scores in SNR, SSIM, RMSE, and MCS. The channels of the filters determine the amount of information that can be detected by CNN. More channels can extract more feature details from the input spectra, and the CNN model can recover the details of the spectra from interferences. To balance the performance of the CNN model with PC computing power and time cost, 64 channels were applied to the CNN in this work.

The fully connected neural network as a classical neural network was also compared with the CNN model we proposed in this work. The fully connected neural network built here has four fully connected layers and the same number of neurons as the CNN we built. It was constructed using TensorFlow 2.2.0 with GPU acceleration on Python 3.8, running on the same laptop as the CNN. The outputs of the fully connected neural network did not retain the characteristic peaks of the original spectra and mixed the features of baseline drifts and noises to the processed spectra (**Figure SI 5**). Another problem is that the training of the fully connected neural network costs much more time than the training of CNN.

### 3.2. *Results for models based on mixed data sets from different samples*

In this paper, we constructed a CNN which aimed to remove the baseline drifts, noises, and spikes in Raman spectra in only one step even when the spectra were collected from samples with far different physical features and chemical compositions. We put the three Raman spectra data sets collected from compressed milk tablets, SARS-CoV-2, and tomatoes together for the training of the CNN model in this section. The simulated spectra were prepared by adding influencing factors (the baseline drifts, noises, and spikes were added; see Fig. 1) to the three data sets and the target spectra were preprocessed by AsLS, SG smooth, and de-spike method. Our CNN model is trained by projecting the simulated spectra with influencing factors to the target spectra processed by the combined methods of AsLS, SG smooth, and de-spike method.

MCS, SNR, RMSE, and SSIM methods were used to assess the performance by comparing the target data and the output data from the CNN model we trained. Similar CNN hyperparameters were used as optimized hyperparameters in the section of CNN model for single datasets: 4 convolution layers, shrinking $1 \times n$ filters, 64 channels, *L2_loss* function, and Adam optimizer.

Fig. 3 illustrates the simulation based on the Raman spectra collected from compressed milk tablet candy, SARS-CoV-2, and tomatoes samples. The simulated interference spectra (red solid line), target spectra (black dotted line), and CNN output spectra (blue solid line) are shown in Fig. 3. Noises, spikes, and baseline drifts can be found in the spectra profiles in the red solid lines. Gaussian noise was added to the spectra and the standard deviation of the noise is 2% of the maximum peak. Cosmic rays are completely random and unpredictable. In this paper, the spikes with random values at random wavelengths were added to every fifty spectra. The Raman spectra of the tomatoes have peaks at 919, 1002, 1150, and 1294 $cm^{-1}$. The Raman spectra of compressed tablet milk candy have peaks at 860, 926, 1082, 1130, 1340, 1463, and 1664 $cm^{-1}$. The Raman spectra of SARS-CoV-2 in wastewater have peaks at 1305, 1436, 1596, 1719, 1743, 1839, and 1861 $cm^{-1}$. The features of the Raman spectra of different samples were far different from each other, and they were used to train the CNN to have a stable and robust model with good generalization. The output spectra are from the well-trained CNN model and they clearly retained all the characteristic peaks and removed the interferences.

Compared to the spectra of tomatoes and compressed tablet milk candy, the spectra of the wastewater samples with SARS-CoV-2 have much more low-intensity peaks, especially in the range of 1700–1800 $cm^{-1}$ (see Fig. 3 **(b))**, and the noise sometimes covers low-intensity signals. CNN gives chance to resolve the pure spectra without noise even when the spectra only provide subtle signals mixed with a high level of noise.

Spikes are always misrecognized as peaks when their intensity is near to the true peaks. From Fig. 3, it can be seen that sufficient training makes the optimized CNN model correctly remove the spikes even though the spikes randomly appear in the simulated spectra with interferences from noises and baseline drifts simultaneously. The results indicate that the CNN model can distinguish between peaks and spikes.

Fig. 4 shows SNR, SSIM, RMSE, and MCS by comparing the preprocessing output spectra from CNN to the target spectra of the test set. We also evaluated the simulated spectra compared to the target spectra. The histogram was displayed for the single dataset of the tomatoes, compressed tablet milk candy, wastewater samples with SARS-CoV-2 separately, and their mixed matrices. The height of the histogram

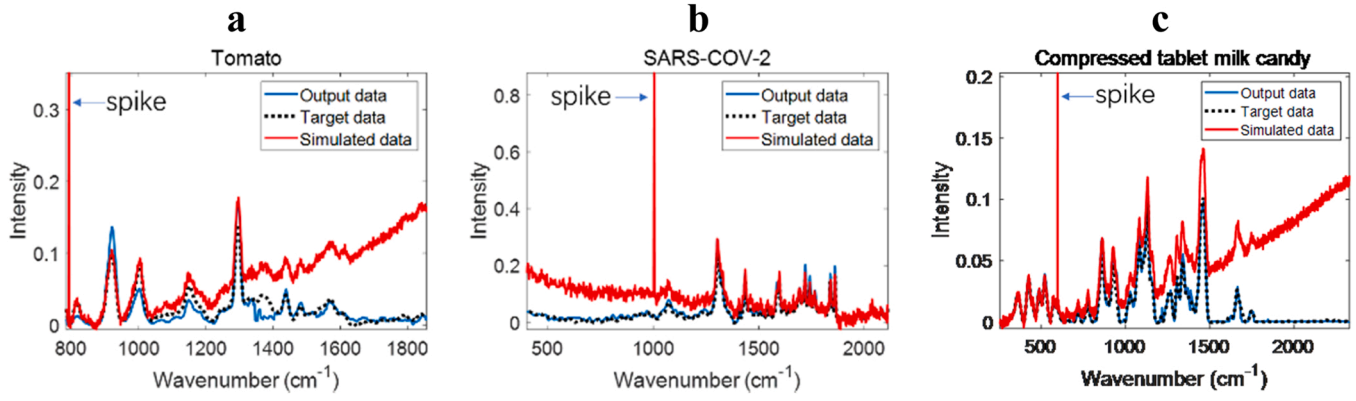


**Fig. 3.** Comparison of CNN outputs (blue solid line), simulated data (red solid line) and target data (black dotted line) which come from three testing data sets, respectively. (a) Examples of Tomato data. (b) Examples of SARS-CoV-2 data. (c) Examples of Compressed tablet milk data.

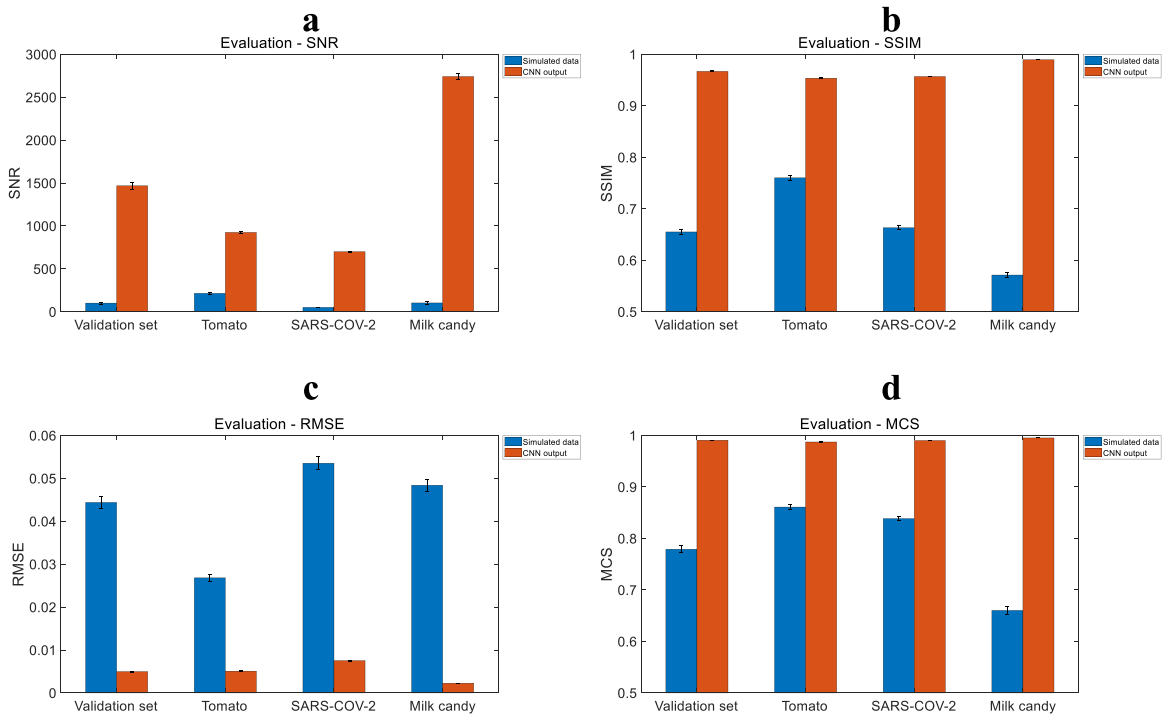


**Fig. 4.** the four evaluation indexes applied to assess the CNN outputs of validation set and testing data sets. (a) SSIM. (b) RMSE (c) SNR (d) MCS.

shows the average of SNR, SSIM, RMSE, and MCS calculated for each spectrum in the test set and the error bars indicate the 95% confidence intervals. The simulated data with interferences (blue bars) and the CNN outputs (red bars) are compared in Fig. 4.

The comparisons of the CNN outputs to the target spectra give higher SNR, SSIM, and MCS and lower RMSE compared to those of simulated data with interference. The average values of SSIM and MCS of the CNN outputs are near 1.0, indicating the CNN outputs are very similar to the target spectra. The SNR of the outputs is far higher than those of the simulated data. The low RMSE represents the error between the outputs and the target data is small.

The scores of the first three components obtained by PCA (Principal Component Analysis) are displayed in **Figure SI 6**. The samples in red color are from the tomato dataset, in blue color are from SARS-CoV-2 water samples, in green color are from milk tablet samples. Three datasets give different distributions when they are not suffering the interferences from baseline, noises, and spikes. When the high level of baseline, noises, and spikes are added, the scatter points get closer in the distribution in **Figure SI 6b**. The Tomato samples are collected from frostbitten and intact tomatoes and they can be clearly clustered in the processed data and the output data from CNN. From the results, we may find that the distance between the samples of the clean data processed by traditional methods is similar to the output spectra from the CNN model.

### 3.3. *The features captured by the hidden layers*

To display the spectral features captured by the neural of the CNN model can help to explain the physical and chemical meaning of neurons in hidden layers. The hidden layers in ANN or CNN models are always considered as a black box in many works of literature. We can visualize the feature maps of hidden layers to display the spectral features extracted from hidden layers[43].

The interpretation of hidden layer of CNN is always challenging. In the published literatures about the application of CNN on the computer vision, which is the most advanced area in deep learning, the feature of imaging captured by hidden layers of deep learning models are commonly displayed using pixel map. The pixel map can provide abstract pattern extracted from the complex image. In some degree, we just want to display the phenomenon in hidden layers of CNN, and we can visualize those features in an intuitionistic way, on which there is still no such objective evolution methods until now. The features in the figures we provided can give apparent shape of the profiles and some of them are coincidently correlate to the baselines, noise and spikes of spectra without the preprocessing.

Although the physical and chemical meaning of CNN is not quite clear, the features of baselines, noise and spikes are easily be observed by visual inspection, which is also generally used in computer vision for checking the resolution of the output and visualized hidden layers of deep learning models.

Fig. 5 gives all features in hidden layers when input one spectrum from compressed tablets milk data set to the CNN model. Fig. 5 **(a1, b1, c1, g1)** are output from convolution layers 1 – 4, Fig. 5 **(a2, b2, c2, d2, e2, f2, g2)** are output from activation layer 1 – 7, Fig. 5 **(a3, b3, c3)** are output from max-pooling layers 1 – 3, and Fig. 5 **(d1, e1, f1)** are output from transposed convolution layers 1 – 3, respectively.

From the convolution layer 1 with a filter of size $1 \times 11$ with 64 channels, a matrix of 4000 (corresponding to spectral direction) $\times$ 64 (corresponding to features of the spectrum) is exported, as shown in Fig. 5 **(a1)**. The output features of layer 1 give the shape of peaks, baseline drifts, and spikes, and they are displayed in Fig. 5 (a1) – Fig. 5 (a3). The profiles in different channels have different intensities, but they have very similar spectra shapes. The convolution kernels were initialized randomly so that the output of the convolution layers always have negative values. The activation layers can increase the non-linearity of the data. Negative values were redundant without physical and chemical meaning, and they can be removed after activation layers.

Fig. 5 **(a3)** gives the output from the max pooling layers which reduced the amount of data to speed up the training process. The data was reduced to $2000 \times 64$, meanwhile, the characteristic bands were kept in the output of pooling layers, and the influence of baseline drifts is reduced in the max-pooling layer 1. In the convolution layer 2 and the activation layer 2 (Fig. 5 **(b1, b2)**), the profiles obtained by different channels provide more different shapes to capture the features of the characteristic bands and interferences. In Fig. 5 **(c2) and (c3)**, the third

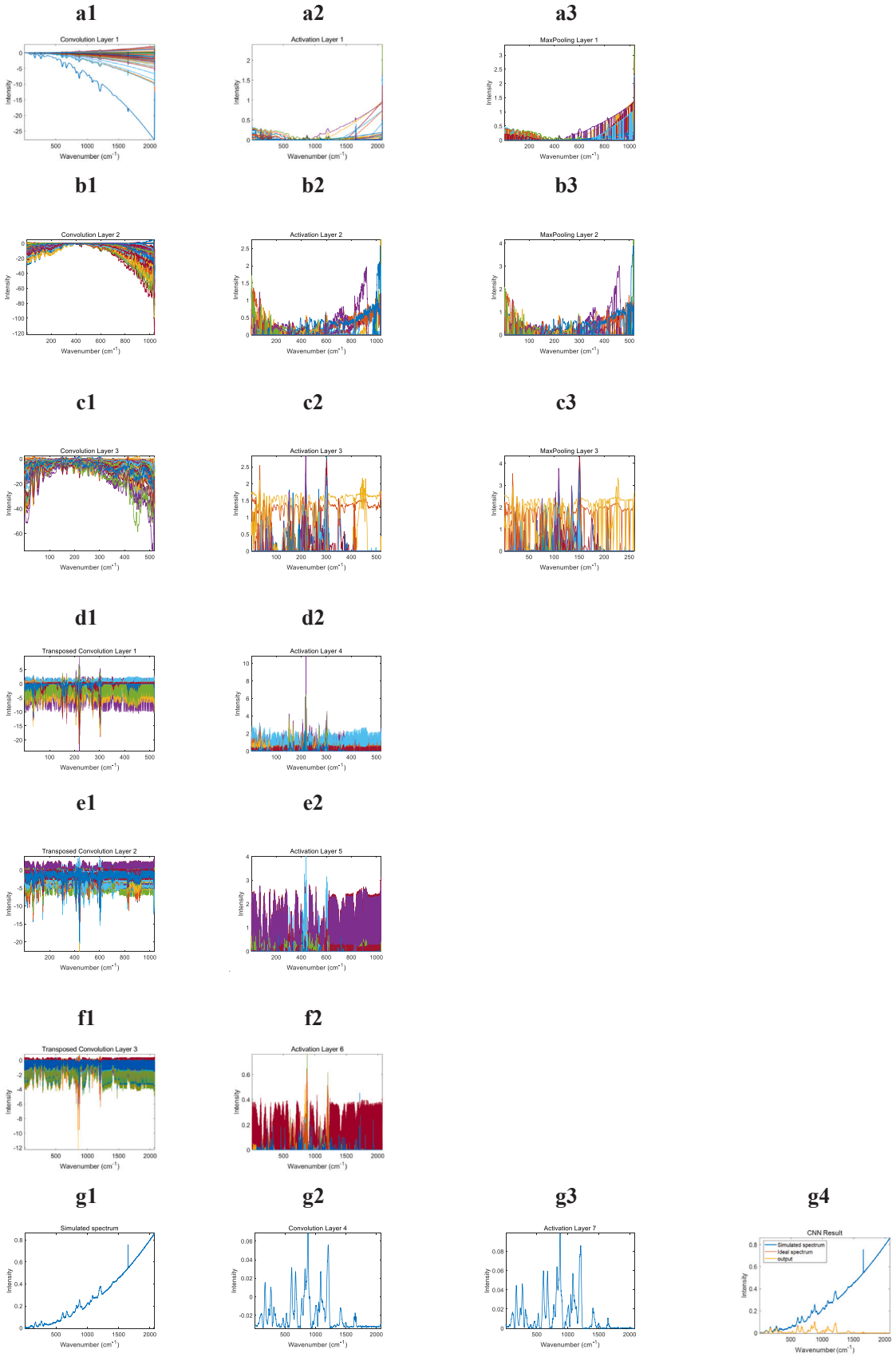


**Fig. 5.** Visualization of the features extracted in the hidden layers using a compressed tablet milk candy spectrum as an example. (a-c) show the features in convolution layer 1, 2, 3, activation layer 1, 2, 3 and max-pooling layer 1, 2, 3, respectively. (d-f) shows the features of transposed convolution layer 1, 2, 3 and activation layer 4, 5, 6, respectively. (g) shows the simulated spectrum, features in convolution layer 4 and activation layer 7 and the comparison of simulated spectrum, ideal spectrum and output.

convolution section outputs some profiles removed baseline drifts.

The output spectra matrix of the max-pooling layer 3 is 500 × 64. The transposed convolution layers recover the size of the spectra matrix to 4000 × 64. In the transposed convolution layer 1 (Fig. 5 **(d1)**), it is manifest that the highest peak of this spectrum is retained, and other lower peaks are buried in random noises. In Fig. 5 **(d-f)**, the shape of the profiles was changing gradually into its corresponding target spectra in the transposed convolution layers. The last convolution layer filters the low-level noises and transform the channel from 64 to 1. The activation layer 7 eliminates the negative values. In Fig. 5 **(g4),** the simulated spectrum with interferences, the target spectrum, and the output spectrum are compared. The output spectrum was well preprocessed to a target spectrum without suffering from noises, baseline drifts, and cosmic rays.

## 4. Conclusion

We developed a deep CNN model and removed all the interferences in Raman spectra data with very different features collected from samples including compressed milk tablets, wastewater with SARS-CoV-2, and tomatoes data sets. This method can handle the influential factors including noise, spikes and baseline in Raman spectra in only one step, and it is compatible with different Raman spectra data set which possesses different numbers of variables. The output of CNN is almost equivalent to the target spectra obtained by integrating cosmic ray removal, AsLS, and SG smooth methods. The MCS value is up to 0.9900. Compared to classical preprocessing methods, the CNN simplifies the preprocessing steps and reduces the cost of time to achieve target spectra. The preprocessing of the drift of Raman shift between instruments is still not included in this work. The presently proposed CNN model-based deep learning method can be easily extended to other spectra preprocessing, and further studies would be made to more complex situations.

## Funding

This work was supported by the National Natural Science Foundation of China (Grant ID: 21705112), China; Capacity Building for Sci-Tech Innovation - Fundamental Scientific Research Funds (Grant ID: KM201910028014), China; Beijing outstanding talents training fund for youth backbone individual project (Grant ID: 2017000020124G081), China.

## CRediT authorship contribution statement

**Jiahao Shen:** Original draft preparation, Methodology, Software, Writing. **Miao Li:** Validation, Reviewing. **Zhongfeng Li:** Investigation, Reviewing. **Zhuoyong Zhang:** Reviewing. **Xin Zhang:** Conceptualization, Validation, Writing – review & editing.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.vibspec.2022.103391.

## References

[1] B. Nagy, A. Farkas, M. Gyurkes, S. Komaromy-Hiller, B. Demuth, B. Szabo, D. Nusser, E. Borbas, G. Marosi, Z.K. Nagy, In-line Raman spectroscopic monitoring and feedback control of a continuous twin-screw pharmaceutical powder blending and tableting process, Int J. Pharm. 530 (2017) 21–29.
[2] K. Chrabaszcz, K. Kochan, A. Fedorowicz, A. Jasztal, E. Buczek, L.S. Leslie, R. Bhargava, K. Malek, S. Chlopicki, K.M. Marzec, FT-IR- and Raman-based biochemical profiling of the early stage of pulmonary metastasis of breast cancer in mice, Analyst 143 (2018) 2042–2050.
[3] A.M. Herrero, Raman spectroscopy a promising technique for quality assessment of meat and fish: A review, Food Chem. 107 (2008) 1642–1651.
[4] S. Goldrick, C.A. Duran-Villalobos, K. Jankauskas, D. Lovett, S.S. Farid, B. Lennox, Modern day monitoring and control challenges outlined on an industrial-scale benchmark fermentation process, Comput. Chem. Eng. 130 (2019), 106471.
[5] P. Mishra, A. Biancolillo, J.M. Roger, F. Marini, D.N. Rutledge, New data preprocessing trends based on ensemble of multiple preprocessing techniques, TrAC Trends Anal. Chem. 132 (2020), 116045.
[6] A. Savitzky, M.J.E. Golay, Smoothing and differentiation of data by simplified least squares procedures, Anal. Chem. 36 (1964), 1627-&.
[7] J. Peng, S. Peng, A. Jiang, J. Wei, C. Li, J. Tan, Asymmetric least squares for multiple spectra baseline correction, Anal. Chim. Acta 683 (2010) 63–68.
[8] V.J. Barclay, R.F. Bonner, I.P. Hamilton, Application of wavelet transforms to experimental spectra: smoothing, denoising, and data set compression, Anal. Chem. 69 (1997) 78–90.
[9] C.S. Cai, P.D. Harrington, Different discrete wavelet transforms applied to denoising analytical data, J. Chem. Inf. Comput. Sci. 38 (1998) 1161–1170.
[10] C. Zhang, F. Liu, Y. He, Identification of coffee bean varieties using hyperspectral imaging: influence of preprocessing methods and pixel-wise spectra analysis, Sci. Rep. 8 (2018) 2166.
[11] A. Zeiler, R. Faltermeier, I.R. Keck, A.M. Tome, C.G. Puntonet, E.W. Lang, Empirical Mode Decomposition - an introduction, The 2010 International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1–8.
[12] P.J. Cadusch, M.M. Hlaing, S.A. Wade, S.L. McArthur, P.R. Stoddart, Improved methods for fluorescence background subtraction from Raman spectra, J. Raman Spectrosc. 44 (2013) 1587–1595.
[13] J. Liu, J. Sun, X. Huang, G. Li, B. Liu, Goldindec: a Novel Algorithm for Raman Spectrum Baseline Correction, Appl. Spectrosc. 69 (2015) 834–842.
[14] S.-J. Baek, A. Park, Y.-J. Ahn, J. Choo, Baseline correction using asymmetrically reweighted penalized least squares smoothing, Analyst 140 (2015) 250–257.
[15] Z.-M. Zhang, S. Chen, Y.-Z. Liang, Baseline correction using adaptive iteratively reweighted penalized least squares, Analyst 135 (2010) 1138–1146.
[16] Y. Xie, L. Yang, X. Sun, D. Wu, Q. Chen, Y. Zeng, G. Liu, An auto-adaptive background subtraction method for Raman spectra, Spectrochim. Acta Part A: Mol. Biomol. Spectrosc. 161 (2016) 58–63.
[17] Y. Liu, W. Cai, X. Shao, Intelligent background correction using an adaptive lifting wavelet, Chemom. Intell. Lab. Syst. 125 (2013) 11–17.
[18] Z. Li, D.-J. Zhan, J.-J. Wang, J. Huang, Q.-S. Xu, Z.-M. Zhang, Y.-B. Zheng, Y.-Z. Liang, H. Wang, Morphological weighted penalized least squares for background correction, Analyst 138 (2013) 4483–4492.
[19] P.H.C. Eilers, A. Perfect, Smoother, Anal. Chem. 75 (2003) 3631–3636.
[20] J. Acquarelli, T. van Laarhoven, J. Gerretzen, T.N. Tran, L.M.C. Buydens, E. Marchiori, Convolutional neural networks for vibrational spectroscopic data analysis, Anal. Chim. Acta 954 (2017) 22–31.
[21] X. Zhang, T. Lin, J. Xu, X. Luo, Y. Ying, DeepSpectra: an end-to-end deep learning approach for quantitative spectral analysis, Anal. Chim. Acta 1058 (2019) 48–57.
[22] J. Wahl, M. Sjödahl, K. Ramser, Single-step preprocessing of raman spectra using convolutional neural networks, Appl. Spectrosc. 74 (2020) 427–438.
[23] L. Ju, A. Lyu, H. Hao, W. Shen, H. Cui, Deep learning-assisted three-dimensional fluorescence difference spectroscopy for identification and semiquantification of illicit drugs in biofluids, Anal. Chem. 91 (2019) 9343–9347.
[24] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186.
[25] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[26] C.M. Valensise, A. Giuseppi, F. Vernuccio, A.Dl Cadena, G. Cerullo, D. Polli, Removing non-resonant background from CARS spectra via deep learning, APL Photonics 5 (2020), 061305.
[27] J. Zupan, J. Gasteiger, Neural networks: a new method for solving chemical problems or just a passing phase? Anal. Chim. Acta 248 (1991) 1–30.
[28] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Curran Associates Inc., Lake Tahoe, Nevada, 2012, pp. 1097–1105.
[29] X.Z. Kaiming He, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, 2015.
[30] J. Yang, J. Xu, X. Zhang, C. Wu, T. Lin, Y. Ying, Deep learning for vibrational spectral analysis: recent progress and a practical guide, Anal. Chim. Acta 1081 (2019) 6–17.
[31] Y. Liu, S. Zhou, W. Han, W. Liu, Z. Qiu, C. Li, Convolutional neural network for hyperspectral data analysis and effective wavelengths selection, Anal. Chim. Acta 1086 (2019) 46–54.
[32] M. Fukuhara, K. Fujiwara, Y. Maruyama, H. Itoh, Feature visualization of Raman spectrum analysis with deep convolutional neural network, Anal. Chim. Acta 1087 (2019) 11–19.
[33] W. Lu, X. Chen, L. Wang, H. Li, Y.V. Fu, Combination of an artificial intelligence approach and laser tweezers raman spectroscopy for microbial identification, Anal. Chem. 92 (2020) 6288–6296.
[34] X. Fan, W. Ming, H. Zeng, Z. Zhang, H. Lu, Deep learning-based component identification for the Raman spectra of mixtures, Analyst 144 (2019) 1789–1798.

[35] Y. Zhang, Raman spectra of compressed milk tablet candy, with calcium and vitamin A,D additives, mendeley, Data V2 (2021).
[36] D. Zhang, X. Zhang, R. Ma, S. Deng, X. Wang, X. Wang, X. Zhang, X. Huang, Y. Liu, G. Li, J. Qu, Y. Zhu, J. Li, Ultra-fast and onsite interrogation of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) in waters via surface enhanced Raman scattering (SERS), Water Res. 200 (2021), 117243.
[37] V. Dumoulin, F. Visin, A Guide convolution Arith. Deep Learn., arXiv Prepr. arXiv 1603 (2016) 07285.
[38] D.P.K.a.J. Ba, Adam: A Method for Stochastic Optimization, 2014.
[39] E.H. John Duchi, Yoram Singer, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
[40] M.C. Mukkamala, M. Hein, Variants of RMSProp and Adagrad with Logarithmic Regret Bounds, in: P. Doina, T. Yee Whye (Eds.) Proceedings of the 34th International Conference on Machine Learning, PMLR, Proceedings of Machine Learning Research, 2017, pp. 2545–2553.
[41] D.Z. Alain Horé, Image quality metrics: PSNR vs. SSIM, 20th International Conference on Pattern Recognition, ICPR 2010Istanbul, Turkey, 23–26 August 2010, 2010.
[42] E.P.S.a.A.C.B. Zhou Wang, Multiscale structural similarity for image quality assessment, The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003Pacific Grove, CA, USA, 2003.
[43] X. Zhang, J. Xu, J. Yang, L. Chen, H. Zhou, X. Liu, H. Li, T. Lin, Y. Ying, Understanding the learning mechanism of convolutional neural networks in spectral analysis, Anal. Chim. Acta 1119 (2020) 41–51.