Application of deep learning in stimulated Raman scattering (SRS) spectra classification

Yifeng Zhou, [1,*]

[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana 47907, USA

*zhou624@purdue.edu

Other members: Peng Lin (lin735@purdue.edu) and Bowen Wei (wei209@purdue.edu)

Abstract: Identification of Raman spectra has been addressed by a variety of methods such as random forest[1], gradient boosting[2] and support vector machine[3]. However, many of them are sensitive to noise, baseline and other imperfection in Raman spectra. Deep learning algorithm has also been applied to this field and shown superior accuracy and efficiency over other methods[4,5]. In this report, we will introduce our deep learning classification algorithm written in Pytorch and discuss its performance and specification.

Our neural network is illustrated in Fig. 1. The Raman volume data is a 400 × 400 × 60 image stack. The neural network will first pickup a point in the xy plane. Then for each pixel, along the z axis, we can get a 1D array containing 60 elements. The neural network will first do 1D convolution and then transform to fully-connected layer. In the end, there will be three outputs identifying the classification of each array. During the experiments, our Raman system acquires images of polymethylmethacrylate (PMMA) and polystyrene (PS) beads. Fig. 2(e) shows the spectra of PMMA and PS. Fig. 2 (a-d) visualize the input and output. Fig. 2 (a-b) are the 400 × 400 × 60 image stack acquired by SRS microscopy, and Fig. 2 (c-d) are the corresponding classification maps, respectively.

In order to optimize the performance of a neural network, during the implementation of neural network, we were always wondering how many neurons are supposed to be in the hidden layer. So we did an investigation of the influence of neuron number on training time and percent error. Fig. 3 (a) shows the training time as a function of the neuron number. We can see that, as long as the neuron number is not extremely large, the training time doesn't change too much. And the error bar is the standard deviation becasue the initialization of a neural network is random, so we need to repeat the training many times, such as 50 times, in order to get an average value and standard

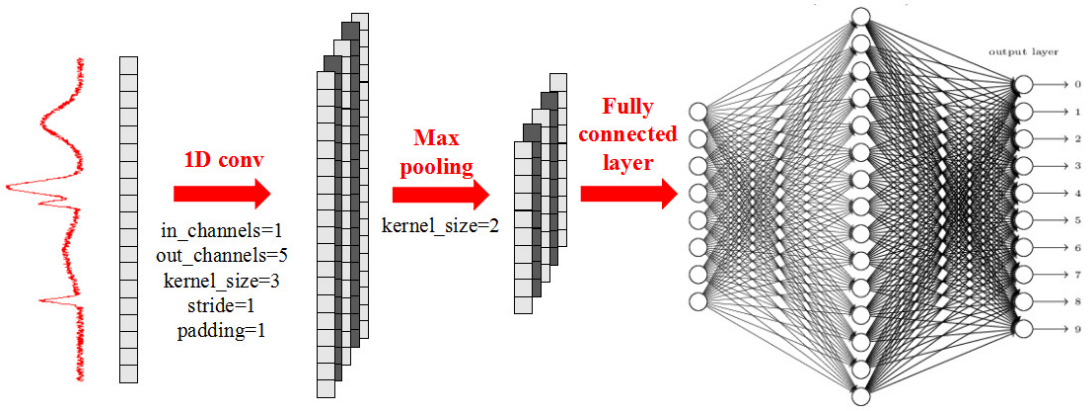

Fig. 1 Illustration of neural network implemented in Pytorch

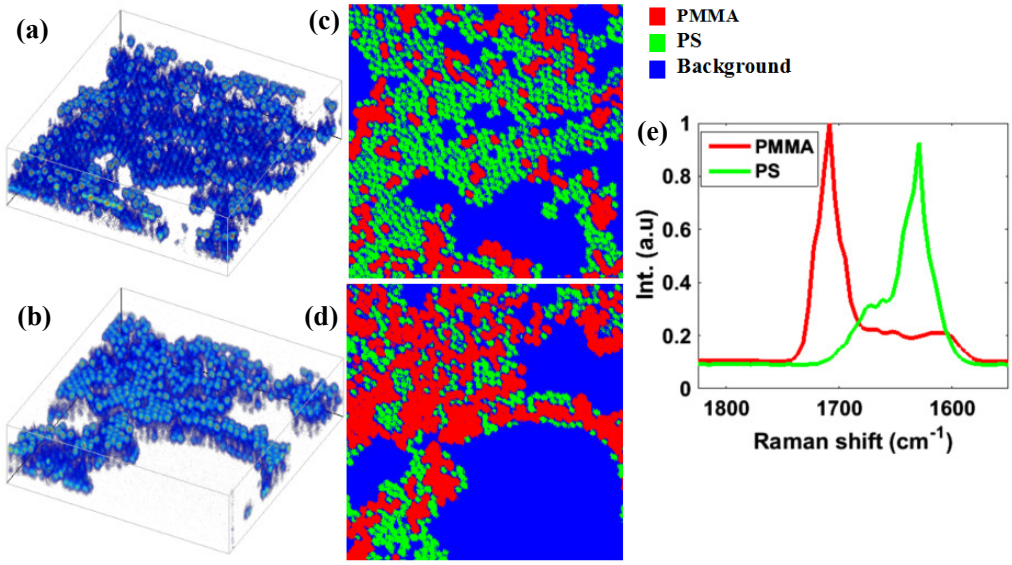

Fig. 2 (a-b) Examples of 3D rendering of Raman volume data. (c-d) Classification maps of (a) and (b), respectively. Red pixel indicates PMMA, green pixel indicates PS and blue pixel indicates background. (e) Raman spectra of PMMA and PS beads

deviation. We can see that if the neuron number is extremely large, such as 2000 or even 10000, the training would be really time-consuming. Of course, in most of the case we will never use such a large number. In addition, we can see that the minimum neuron number in Fig. 3 (a) is 20 because we once tried to set the neuron number to 10, however in this case, the neural network is always unable to converge. Besides, Fig. 3 (b) shows the percent error as a function of neuron number. We can see that the percent error slightly decreases as the neuron number increases. But

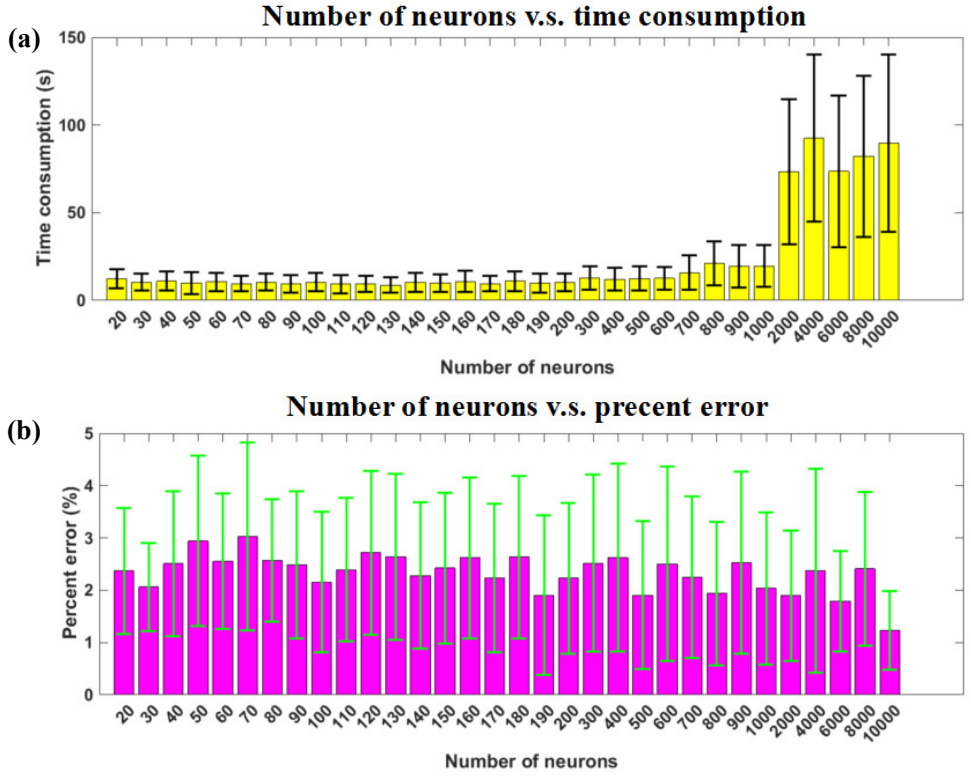

Fig. 3 (a) Training time as a function of number of neurons (b) Percent error as a function of number of neurons. Scale bar represents the standard deviation of 50 repetitions.

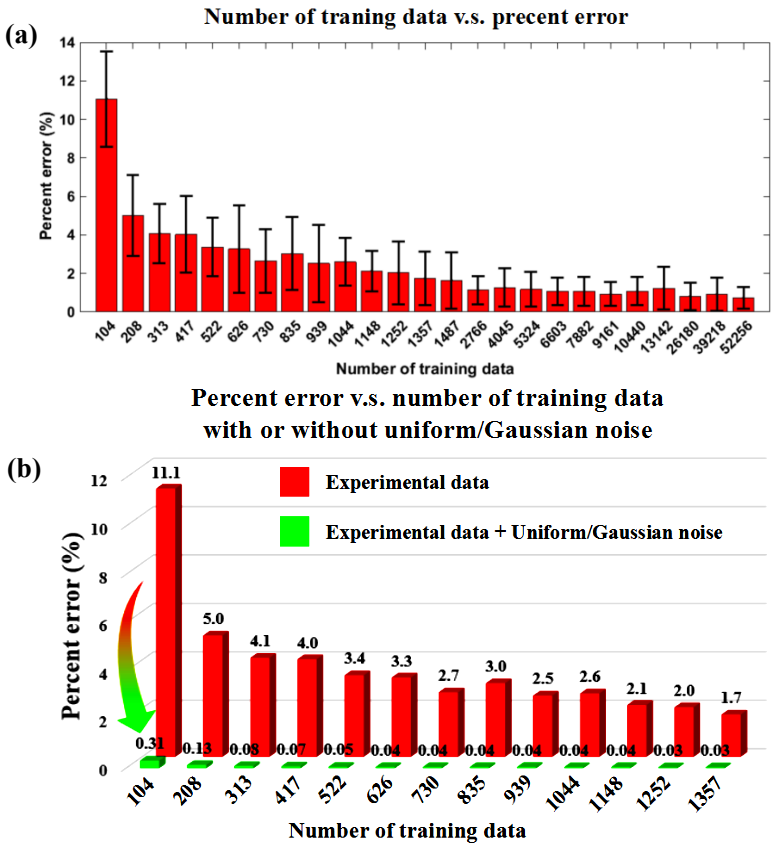

Fig. 4 (a) Percent error as a function of number of training data. (b) Percent error as a function of number of training data with/without artificial uniform/Gaussian noise. The post-introduced noise increases the number of training data by a factor of 60

overall, the change of neuron number has no significant influence on percent error. So from Fig. 3, we can conclude that as long as the neuron number is not extremely small or extremely large, the performance of neural network is quite stable. As a result, we will usually set the neuron number to 30 during the later implementation of neural network.

The next question is that, during the training, we all hope to have as much data as possible in order to train the neural network well. Then Fig. 4 (a) shows the percent error as a function of training data. As expected, the percent error decreases as the number of training data increases. Besides, we can also observe that as long as the number of training data is roughly larger than 3000, the percent error will drop to about 1 or 2% and no longer decrease significantly. And in general, 3000 training data is a reasonable number and it is not hard to acquire it during the imaging experiments. Besides, on the left hand side we can see that if we only have 100 training data, the error is about 11%, which is too large to be applied in practice. But actually it is not strange to get stuck in this problem because sometimes there may be something wrong with the imaging experiments so we have very limited data. Then Fig. 4 (b) shows us the answer. The red column is the percent error as shown in Fig. 4 (a). Then we tried to introduce uniform and Gaussian noise to the raw in order to increase the number of training data by a factor of 60. As shown by the green column, we can see that even if there are only 100 training raw data, the error can be dramatically reduced from 11% to 0.3%, which is 35 times smaller than before. And the percent error can be further reduced if we can provide more and more data. This is a great news because it is really easy for us to acquire

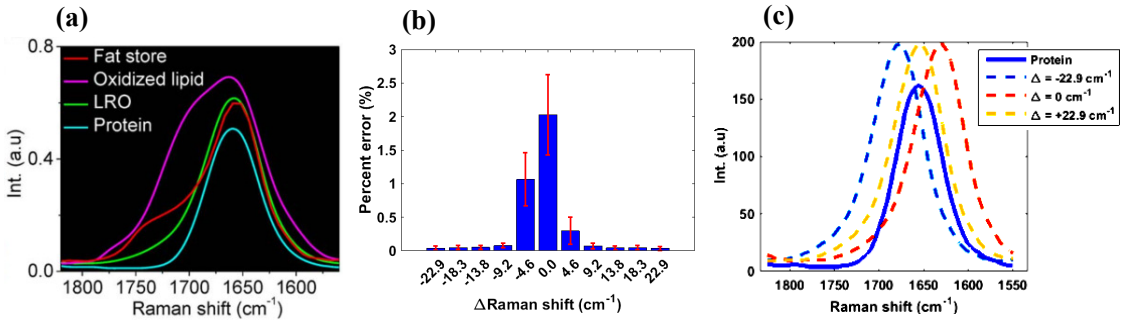

Fig. 5 (a) Raman spectra of fat store, lipid, LRO and protein[6]. (b) Percent error as a function of Raman shift, the error bar represents the standard deviation of 50 repetitions. (c) Raman spectra of protein and shifted LRO.

only 100 raw data during the imaging experiment and the corresponding accuracy, 99.7%, is more than enough to be applied in practice. So here we can conclude that, with the help of post-introduced noise, we can achieve much higher accuracy even if the raw data is much less than before.

Fig. 5 (a) shows the Raman spectra of fat store, lipid, LRO and protein[6]. We can observe that the location of their peaks are close to each other compared to PMMA and PS. Especially, we can see that the spectra of LRO and protein are really similar to each other, the main difference is that they have different amplitude. Intuitively, it would be difficult for the neural network to distinguish overlapped spectra. So here we tried to quantify the influence of overlap. However, currently we don't have experimental raw data of LRO and protein at hand so we simply use the data in the reference paper[6] and add artificial noise to form training data. Fig. 5 (b) shows the percent error as a function of Raman shift. We tried to shift the LRO spectrum manually and feed it to neural network and calculate the corresponding percent error as shown in Fig. 5 (c). Unsurprisingly, the percent error reaches its peak when the spectra of LRO and protein are highly overlapped, which is about 2%. If the LRO spectrum is shifted far away, the percent error can be decreased to ~0.03%, similar to PMMA and PS case. So here we can see that overlapped spectra will greatly increase the percent error. But such a phenomenon is inevitable because the spectrum of a material is determined by nature. In the future, we will try to use new methods to address this issue.

References:

1. Ho, Tin Kam. "The random subspace method for constructing decision forests." IEEE transactions on pattern analysis and machine intelligence 20, no. 8 (1998): 832-844.

2. LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86, no. 11 (1998): 2278-2324.

3. Vapnik, Vladimir. The nature of statistical learning theory. Springer science & business media, 2013.

4. Liu, Jinchao, et al. "Deep convolutional neural networks for Raman spectrum recognition: a unified solution." Analyst 142.21 (2017): 4067-4074.

5. Manescu, Petru, et al. "Accurate and interpretable classification of microspectroscopy pixels using artificial neural networks." Medical image analysis 37 (2017): 37-45.

6. Liao, Chien-Sheng, et al. "Stimulated Raman spectroscopic imaging by microsecond delay-line tuning." Optica 3.12 (2016): 1377-1380.

Author contribution:
Yifeng Zhou built up the neural network, got the classification maps and investigated the training time and percent error as a function of number of neurons. Peng Lin acquired the PMMA/PS raw data using Raman spectroscope and investigated the percent error as a function of number of raw data. Bowen Wei assisted the Raman imaging and tried to figure out the percent error as a function of Raman shift. All group members will try to transfer Window-based hardware and software to Linux operating system and verify if it can classify Raman raw data accurately and in real-time.

Code & raw data: https://github.com/zhou624/BME595_finalreport
Slide link: https://drive.google.com/open?id=1py-wzJACI0AEX9TRIUZxmhd0W9XsMRQo
YouTube video: https://youtu.be/1gvikNmgnNU