Classification of Cannabinoid Spectra Using Machine Learning

Narayan Kabra¹, Gianluca Serra³, Miguel Lozano², Matteo Tommasini³, Karthik Sankaranarayanan², and Nisha Rani Agarwal*¹

¹ Nano-imaging and Spectroscopy Laboratory, Faculty of Science, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa ON L1G0C5, Canada
² Faculty of Business and Information Technology, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa ON L1G0C5, Canada
³ Department of Chemistry, Chemical and Materials Engineering ‘Giulio Natta’, Politecnico di Milano, Piazza Leonardo da Vinci, 32, Milan 20133, Italy

*Corresponding author: nisha.agarwal@ontariotechu.ca

Abstract

Vibrational spectroscopy, encompassing Raman and Infrared (IR) spectroscopy, is a powerful technique that probes the intrinsic vibrations of a molecule, thus providing a unique chemical signature for that molecule. This information is beneficial to differentiate between two similarly structured molecules since their vibrational fingerprint will be different. In an effort to introduce an automated spectroscopic data analysis tool, we explore different Machine Learning (ML) algorithms to identify the chemical structure from the simulated Raman and IR spectra of 22 similar molecules belonging to the class of cannabinoids.

In this study, we investigate the best ML approach by using representative synthetic IR/Raman data obtained from quantum chemical calculations of the selected molecular structures. We account for the experimental variability of the spectra by adding different kinds of noise and backgrounds to the simulated spectra such that they mimic experimental conditions such as fluorescence background as well as Gaussian noise. This methodology
is used to setup the database to train the ML algorithms. We report the accuracy of the
different ML algorithms and the time taken to process the algorithms in differentiating the
cannabinoid varieties.

**Keywords** — Machine Learning, Vibrational Spectroscopy, Raman, Infrared, Cannabinoids,
Spectral classification

1 Introduction

Raman and infrared (IR) spectroscopies are powerful techniques which allow identifying the
chemical-specific fingerprint of a molecule without the need for labelling and with minimum sample
preparation [1, 2]. This is a considerable advantage to chemists and biologists for the recognition
and characterization of synthesized drugs and chemicals, and assessing the purity of compounds
[3]. Moreover, Raman and IR spectroscopies are complementary techniques, as they differ in
the selection rules for the different vibrational modes. That is, Raman spectroscopy detects
those vibrational modes associated with changes in polarizability whereas in IR spectroscopy,
vibrations associated to changes in dipole moment are registered [4]. Both techniques are
non-invasive, and minimum preparation of sample is required for them [2]. Additionally, these
techniques are versatile since they can characterize the vibrational signature of different types
of materials ranging from chemical compounds to biological samples [5]. Both these techniques
have advantageous features which can be put to use. For instance, compounds in aqueous
environments and moist samples can be easily measured in Raman spectroscopy since water
does not interfere with Raman spectral measurements. Furthermore, Raman spectroscopy allows
direct measurement and material identification through plastic bags and glass vials, reducing the
chance of contamination.

Both benchtop and portable Fourier transform IR (FTIR) and Raman spectrometers are
commercially available. However, ease of accessibility and transportation, as well as providing
onsite and almost instantaneous results, has gained considerable amount of attention for portable
instruments among researchers. Moreover, portable units have specific forensic applications such
as the identification of street drugs, and explosive materials, which requires rapid and accurate
analysis. It is also beneficial for field applications such as quality control of pharmaceutical drugs [6].

Despite having unique features, characteristics and ease of access, these techniques are not widely used since the output of vibrational spectroscopy is a complex spectrum with peaks and valleys. Such spectrum can only be read by expert researchers who understand the physics behind the vibrational modes and selection theory, thus gaining insight into the vibrational features of the sample. This limits the employability of vibrational spectroscopy to a few research laboratories and industries despite its beneficial performance and numerous capabilities. In this paper, we provide a solution to this limitation by using machine learning (ML) as a tool to obtain an intelligent vibrational spectroscopy system that identifies the different cannabinoids.

ML refers to algorithms that enable computers to learn complex patterns from data, usually those that wouldn’t be possible to model with other multivariate techniques. ML arose in 1959 with the developments made by Arthur Samuel, who was an authority in the field of artificial intelligence and computer gaming [7].

The general workflow of an ML algorithm goes as follows: there is a training set that contains the ground truth of the variable that is the object of prediction, as well as features that will serve as an input to generate that prediction [8]. Supervised learning refers to those ML models where there is a ground truth on what the user is trying to predict, in other words, there’s a value for a dependent variable, a label for classification, or something of similar nature, that would allow the ML algorithm to determine how good it’s predictions are and fine tune its parameters to minimize the error. Additional examples on cases where supervised learning has proved its usefulness are: classification problems, such as identity fraud detection, image classification, object detection, medical diagnostics, and regression problems [9–12].

Machine learning has been widely used in numerous fields such as healthcare [13–17], agriculture [18–22], robotics [23–27], finance [28–32], security [33–37], and natural language processing [38–42]. It has been particularly successful in prediction, decision making, and modelling of complex problems with higher dimensionality, where it has surpassed the performance of some humans, sometimes even experts. Computers have beaten humans at single character recognition in reading based human interaction proofs [43]; Google’s AlphaGo beat the best Go player in the
world Lee Se-dol multiple times; IBM’s Deep Blue algorithm beat Garry Kasparov, the best chess player at the time; convolutional neural networks have proven to have less false positives than an average human pathologist at detecting metastasis in tumors as small as 100x100 pixels in gigapixel microscopy images sized 100,000x100,000 pixels.

The works by Lansford et al. and Kananenka et al. demonstrate the successful use of ML based approaches to infer from vibrational spectroscopy data details about the microstructure of systems of interest, such as solvation shells and catalytic surfaces. Other ML approaches on spectroscopic data have been used with success to differentiate different samples based on spectral differences.

Ho et al. identify that Raman spectroscopy has numerous challenges, where speeds and accuracies are low due to weak Raman signal from bacterial cells and numerous bacterial species and phenotypes. The authors generate an extensive dataset of bacterial Raman spectra and apply deep learning to accurately identify 30 common bacterial pathogens. On low signal-to-noise spectra, they achieve an average isolate-level accuracy exceeding 82% and antibiotic treatment identification accuracies of 97%. The authors use a convolutional neural network (CNN) as their main model for identification.

The motivation behind this study is to be able to perform chemical analysis on molecules, and to create a system for quick and reliable classification of compounds while also allowing the user to experiment with noise simulation and reproducing the effects of different experimental conditions on the spectrum. In this paper, the wavenumbers and corresponding intensities provided by both Raman and IR spectroscopies are used as inputs for classifying cannabinoid molecules with ML digitizing the classification from these spectral peaks.

2 Materials and Methods

2.1 DFT calculations for generating vibrational spectroscopy data of cannabinoids

Density Functional Theory (DFT) was used to compute the IR and Raman spectra of 22 cannabinoids at the B3LYP/6-31G(d,p) level. The Gaussian09 package was employed.
## Table 1: Sample of ML Training Dataframe

<table>
<thead>
<tr>
<th>Cannabinoid</th>
<th>Label</th>
<th>IR 1</th>
<th>IR 2</th>
<th>...</th>
<th>IR N</th>
<th>R 1</th>
<th>R 2</th>
<th>...</th>
<th>R N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compound 1</td>
<td>1</td>
<td>Intensity</td>
<td>Intensity</td>
<td>...</td>
<td>Intensity</td>
<td>Intensity</td>
<td>Intensity</td>
<td>...</td>
<td>Intensity</td>
</tr>
<tr>
<td>Compound 2</td>
<td>2</td>
<td>Intensity</td>
<td>Intensity</td>
<td>...</td>
<td>Intensity</td>
<td>Intensity</td>
<td>Intensity</td>
<td>...</td>
<td>Intensity</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Compound N</td>
<td>N</td>
<td>Intensity</td>
<td>Intensity</td>
<td>...</td>
<td>Intensity</td>
<td>Intensity</td>
<td>Intensity</td>
<td>...</td>
<td>Intensity</td>
</tr>
</tbody>
</table>

IR = Infrared Wavenumber | R = Raman Wavenumber

### 2.2 Setup of the dataset

The collection of experimental IR/Raman data on cannabinoids is very costly (because of compound volatility). Thus before undertaking this challenge one would like to know if there is a...
chance of success. Hence we simulated by accurate DFT calculations how vibrational spectra
differ from molecule to molecule.

We reproduce experimental conditions by introducing noise into the dataset. The purpose of
noise introduction is two-fold. First, to create multiple samples by generating new observations of
each spectrum. The reason is that ML models require several samples of each class to train and
test [8]. Second, to experiment with different levels and types of noise to replicate experimental
spectra. Noise is an unwanted fluctuation in the signal that can be caused by multiple factors
such as measurement noise, detector dark current noise, and noise due to the background or
environment of the lab [60]. In Raman spectroscopy, the intensity of Raman peaks is very small
compared to the intensity of the intrinsic fluorescence background that appears as a changing
baseline signal [61]. There is also a possibility of broadening the peaks, which makes them harder
to be distinguished.

2.2.1 Gaussian Noise

: We draw samples from a normal (Gaussian) distribution data generator in Python, where the
'average' of the distribution is the intensity itself, and the standard deviation corresponds to the
degree of noise to be introduced into the model. We run a for loop by varying the standard
deviation from 0.001 to 0.05. For each standard deviation value, we append the corresponding
newly generated set of observations to our dataset. An example has been illustrated in Figure 2

2.2.2 Broadening of the peaks/ smoothing of curves

: We smooth the peaks out by applying a 1-D filter called the Savitzky-Golay filter to the dataset
as shown in Fig. 3. We import the 'savgol_filter' from the Signal Processing toolbox of Python
(scipy.signal). The filter accepts the following necessary parameters:

1. The data to be filtered (our original spectra).
2. The length of the filter window.
3. The order of the polynomial used to fit the samples.
We vary the values of the second and third parameters over two separate 'for' loops to obtain different degrees of smoothing. We append the noise-induced dataset to our primary dataset.

2.2.3 Fluorescence background

To add this type of noise (example shown in fig.4), we follow these steps:

1. We create a NumPy array of wavenumbers and append it to itself so that it can correspond to the columns of both IR and Raman peaks.

2. We then create a pandas series called 'baseline' as a function of the wavenumbers array. We choose a sinusoidal function to replicate the fluorescent background signal.

3. By running a 'for' loop, we add the 'baseline' to each dataset row (since each row corresponds to a sample) while simultaneously reducing the magnitude of the spectrum intensity peaks.

4. For creating multiple samples, we run another loop over the previous one by varying the reduction of the peaks.

2.3 Machine Learning setup

The ML scripts have been done in a Google Colab Notebook written in Python (version 3.7). The following libraries have been used: Tensorflow (version 2.8.2), Keras (version 2.8.0), NumPy (version 1.21.6), and Scikit-learn (version 1.0.2). Google Colab’s GPU and environment have been used to process the ML algorithms.

2.3.1 Training and testing ML models:

First, we split the data-set into three parts, in the following ratio:

- 70% training set
- 15% validation set
- 15% test set
We train the following ML models after importing the corresponding algorithms from the scikit-learn library of Python:

- **Logistic Regression**: We use multinomial logistic regression when our outcome variable is nominal and the categories to be predicted do not have an order. We set the 'random_state' parameter (for shuffling the data) to '0' and the 'solver' parameter (the algorithm used for optimization problem) to 'liblinear'. It uses maximum likelihood estimation to evaluate the probability of categorical membership.  

- **K-nearest neighbors**: The KNN algorithm assumes that similar things exist in close proximity to each other. It compares the classes of the 'K' nearest data points to classify the data point given. We let all the parameters have default values.

- **Gaussian Naïve-Bayes**: Naïve Bayes (NB) classifiers are based on Bayes’s theorem. Gaussian NB, a probabilistic classifier, assumes that each class follows a Gaussian distribution. The assumption this model makes is that each feature makes an independent and equal contribution to the outcome. We let the parameters have default values.

- **Decision Tree**: Unlike the probabilistic approach of Gaussian Naive Bayes, the Decision Tree algorithm is rule-based. It uses tree representation where the classes are the leaf nodes and the attributes are the internal nodes. We set the 'random_state' parameter (for shuffling the data) to '0'.

- **Random Forest**: This algorithm consists of a large number of Decision Trees that work together as an ensemble. Each tree makes its prediction, and the one with the highest votes becomes the model’s prediction. We set the 'random_state' parameter (for shuffling the data) to '0'.

We also train a Convolutional neural network (CNN). A CNN is an artificial neural network with different kinds of layers. The convolution layer is used to extract local features, the pooling layer is used for dimensionality reduction to avoid over-fitting, the fully connected layers are used for generating outputs. For our CNN model, we first import the necessary libraries. Our next step is to reshape the 'X' data frames from 2-D to 4-D so we can feed them into the CNN model. We
perform one-hot encoding on the 'Y' variables. This is used for nominal variables, but we keep it in the code for reproducibility. We use Keras’ Sequential model to stack our CNN layers. Its layers are:

- 1st 2D convolution layer. The input shape of our data is [a,b,1] where \( a \times b \) gives the total number of feature columns in our dataset. Our activation function here is ReLu and padding is ‘same’.

- 1st 2D Max Pooling layer. We keep the window size as 2 * 2.

- 1st Dropout layer with the rate 0.25.

- 2nd 2D convolution layer with increased number of filters (output dimension) and decreased kernel size.

- 2nd 2D Max Pooling layer with increased stride.

- 2nd Dropout layer with the same rate.

- 1st Flatten layer

- 1st Dense layer with output dimension of 256 and activation function ‘relu’.

- 3rd Dropout layer with increased rate of 0.5.

- 2nd Dense layer (our final layer) with output dimension the size of our predictions, and a 'softmax' activation function.

We use the Adam optimizer and set the 'loss' parameter for configuring the model to 'categorical_crossentropy'. We set the 'metrics' to 'accuracy', which calculates how often predictions equal labels.

We set epochs to 50, keep the batch size as 250 and set steps per epoch equal to number of input samples. After fitting and evaluating the model, we use the seaborn library for plotting a confusion matrix and wrap up our CNN results with a performance report.
We tabulate our results for each session, wherein each table compares classification accuracies among different algorithms for the same combination of noise in the data. We also compare our results across different combinations of noise.

3 Results and Discussion

We observe from Figure 1D that the vibrational spectrum of the different cannabinoids look very similar to each other from the DFT calculations. After addition of the different noises to the spectrum as seen in Figures 2D, 3D, 4D and 5D, their spectra become even more indistinguishable and hence cannot be discerned by the naked eye. Figures 2A, 3A, 4A and 5A compare the DFT simulated spectrum (clear of all noise) with a noise induced spectrum. One can clearly observe in Figures 2B, 3B, 4B and 5B (spectral range: 1000 to 1800 cm\(^{-1}\)) the alteration of peak signals due to the different noises that have been introduced to the pristine DFT simulated spectrum.

In Tables 2 and 3, we report the Accuracy and Time of the different ML algorithms tested on the cannabinoid spectra. Here, accuracy is given by the "accuracy score" function from scikit-learn library’s metrics module. In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0. Time here, in reference to a model, refers to the approximate time taken to train the model on the given data set. This value may change with different platforms, and on the usage of a GPU vs a CPU. The main difference in the input data between Tables 2 and 3 is that Table 2 contains 432 input spectra (99 Raman spectra with Gaussian noise for each cannabinoid: Fig. 2C, 99 Raman spectra with Smoothing noise for each cannabinoid: Fig. 3C, 18 Raman spectra with Fluorescence background for each cannabinoid: Fig. 4C, 99 IR spectra with Gaussian noise for each cannabinoid, 99 IR spectra with Smoothing noise for each cannabinoid, 18 IR spectra with Fluorescence background for each cannabinoid) and Table 3 contains 200 input spectra (100 Raman spectra with Gaussian + Smoothing + Fluorescence noise for each cannabinoid: Fig. 5C, 100 IR spectra with Gaussian + Smoothing + Fluorescence noise for each cannabinoid).

We draw observations from two different cases.
Table 2: Comparison for Case 1

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Test Accuracy</th>
<th>Time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LR</td>
<td>98.32</td>
</tr>
<tr>
<td>1</td>
<td>KNN</td>
<td>97.73</td>
</tr>
<tr>
<td>2</td>
<td>GaussianNB</td>
<td>17.65</td>
</tr>
<tr>
<td>3</td>
<td>Decision Tree</td>
<td>97.63</td>
</tr>
<tr>
<td>4</td>
<td>Random Forest</td>
<td>99.21</td>
</tr>
<tr>
<td>5</td>
<td>CNN</td>
<td>94.18</td>
</tr>
</tbody>
</table>

Table 3: Comparison for Case 2

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Test Accuracy</th>
<th>Time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LR</td>
<td>89.35</td>
</tr>
<tr>
<td>1</td>
<td>KNN</td>
<td>97.63</td>
</tr>
<tr>
<td>2</td>
<td>GaussianNB</td>
<td>86.39</td>
</tr>
<tr>
<td>3</td>
<td>Decision Tree</td>
<td>95.86</td>
</tr>
<tr>
<td>4</td>
<td>Random Forest</td>
<td>98.22</td>
</tr>
<tr>
<td>5</td>
<td>CNN</td>
<td>2.98</td>
</tr>
</tbody>
</table>

Case 1: We draw separate set of samples from Gaussian distributed data, curve smoothing data and data with fluorescence noise and append them to our original data. We then train all our models on this larger dataset and tabulate the results. We compare the models based on their accuracy of classification of the testing data as shown in Table 2.

Case 2: We apply all kinds of noise on the same data-set as shown in Figure 5. We draw samples from the Gaussian distributed data, smooth the spectral curves, and add a fluorescent signal to them. We then compare them in Table 3 similarly as in Case 1.

On average, the Random Forest Classifier performs consistently well with different combinations of noise in the data. The models perform better in Case 1 than in Case 2. Comparing the two Tables 2 and 3, we observe that when a large number of samples are drawn from a wider Gaussian distribution, Logistic Regression Classifier and CNN perform better than when a lesser
number of samples are drawn from a narrower Gaussian distribution, while the opposite is true for the other models. An explanation lies in the behaviour of the accuracy of CNNs with respect to the number of samples in their training sets. Indeed, ML algorithms become monotonically more accurate with larger datasets [67, 68], with a sharp increase of accuracy in the region with few samples. We thus argue that Cases 1 and 2 lead to drastically different accuracies because Case 2 contain a relatively small amount of spectra, in such a way that even a small increase in their number causes an important difference in the performances of the related CNNs. This is in accordance with other studies using thousands of spectra to train CNNs [51, 69, 70].

In an individual session, we observe that fitting the model takes most time for LR and least for KNN, while their accuracies are comparable with the exception of the last combination of data. Tables 2 and 3 also illustrate how different machine learning models are differently accurate, as it is also found in other studies [71, 72].

4 Conclusion

In this study, we were able to simulate different kinds of noise that can be experimentally found in the spectra obtained using vibrational spectroscopy. We generated multiple sets of samples from our theoretical spectra and were able to use them to train five different ML models and a convolutional neural network. We experimented with different combinations of noise in the data and compared the results of classification of the mentioned algorithms. We were thus able to create a system for chemical analysis by using ML with vibrational spectroscopy, and noise simulation with theoretical data.

On the chemical outcome side, using ML data analysis approaches we can differentiate molecules within the same class, despite the similarities among the spectra and the effect of noise and background. The best performing ML algorithms are KNN and Random Forest which both perform well and are time effective. CNN on the other hand is unpredictable with small datasets.
5 Acknowledgements

The authors would like to thank Mattia Frattini, graduate student of Prof. Matteo Tommasini for preparing the input structures that were used for the DFT calculations of the spectra reported here.

References


[62] Dr. Jon Starkweather and Dr. Amanda Kay Moske. Multinomial Logistic Regression.


Figure 1: DFT simulated Raman spectra of 22 cannabinoids along with their structures
Figure 2: A) Example of simulated Gaussian noise of ‘CBGA’ cannabinoid spectrum on entire spectral range i.e. 0 to 4000 cm\(^{-1}\). B) Example of simulated Gaussian noise of ‘CBGA’ cannabinoid spectrum on part of spectral range i.e. 1000 to 1800 cm\(^{-1}\). C) 3D plot showing all spectra generated with variations in Gaussian noise for ‘CBGA’ cannabinoid as an example. D) One simulated Gaussian noise of the 22 cannabinoids.
Figure 3: A) Example of simulated smoothing noise of 'CBGA' cannabinoid spectrum on entire spectral range i.e. 0 to 4000 cm\(^{-1}\). B) Example of simulated smoothing noise of 'CBGA' cannabinoid spectrum on part of spectral range i.e. 1000 to 1800 cm\(^{-1}\). C) 3D plot showing all spectra generated with variations in smoothing noise for 'CBGA' cannabinoid as an example. D) One simulated smoothing noise of the 22 cannabinoids.
Figure 4: A) Example of simulated fluorescence noise of 'CBGA' cannabinoid spectrum on entire spectral range i.e. 0 to 4000 cm\(^{-1}\). B) Example of simulated fluorescence noise of 'CBGA' cannabinoid spectrum on part of spectral range i.e. 1000 to 1800 cm\(^{-1}\). C) 3D plot showing all spectra generated with variations in fluorescence noise for 'CBGA' cannabinoid as an example. D) One simulated fluorescence noise of the 22 cannabinoids.
Figure 5: A) Example of all the simulated noise (i.e. Gaussian + Smoothing + Fluorescence) of 'CBGA' cannabinoid spectrum on entire spectral range i.e. 0 to 4000 cm\(^{-1}\). B) Example of all the simulated noise of 'CBGA' cannabinoid spectrum on part of spectral range i.e. 1000 to 1800 cm\(^{-1}\). C) 3D plot showing all spectra generated with variations in all the simulated noise for 'CBGA' cannabinoid as an example. D) One simulated noise (Gaussian + Smoothing + Fluorescence) spectrum of the 22 cannabinoids.