Supplementary Information (SI) for Analyst.

Supporting Information

Generative Adversarial Network-driven high-resolution Raman spectral generation for accurate molecular feature recognition

Vikas Yadav,[‡] Abhay Kumar Tiwari,[⊥] Soumik Siddhanta,[‡*]

[‡]*Department of Chemistry, Indian Institute of Technology Delhi, Hauz Khas, New Delhi, India, 110016*

[⊥]*SPATIALTY.AI, Tagya Research Private Limited, Rajankunte, Bangalore 560064*

*Corresponding author e-mail: soumik@iitd.ac.in

Table of Contents

Section S1: Material and Methods

S1.1: Raman spectral acquisition

The LR spectrometer possesses an approximate resolution of 31 cm^{-1}, featuring a grating with 150 lines per millimeter and incorporating a linear CCD array. The collection optics involved an Integrated Photonics System optical fiber probe coupled to a 785 nm laser (model: I0785SP100-T040S). The Raman probe has a central fiber for light input and is surrounded by six fiber endings to collect the scattered light. A ball lens is mounted to focus the light into the sample and collect the reflected light. The focal length of this probe is 4 mm. Raman spectra were acquired from aspirin, ibuprofen, paracetamol, ranitidine, 4-MBA, and 4-NTP. Approximately 400 Raman spectra were collected from each sample at different positions to account for spectral variability. Another set of high-resolution spectra was collected using the high-resolution Andor spectrophotometer (HR spectrometer) (Kymera 328i, with quad turret, 328 mm focal length, F/4.1 aperture having CCD camera with resolution < 1 cm^{-1}, grating 1800 lines/mm) with similar acquisition setting, i.e., using a 785 nm laser with acquisition time of 15 s and three accumulations, and 1800 lines/mm grating system. All Raman spectra were calibrated against the first-order silicon phonon mode at 520.7 cm^{-1} to ensure alignment of the wavenumber axis across both low-resolution and high-resolution datasets. The calibration accuracy was within ±1 cm^{-1}, which is negligible compared to the spectral resolution difference between instruments (31 cm^{-1}). This calibration step ensured that GAN training was not affected by systematic shifts in peak positions, thereby preserving the fidelity of spectral reconstruction. We also accumulated 400 Raman spectra from each sample using the HR spectrometer. A series of spectra were recorded from each sample to create a comprehensive dataset for training the GAN. Care was taken to minimize external interferences, and all measurements were conducted in a controlled environment to maintain consistency. We have accumulated the spectrum from 200 nm to 1650 nm with 862 wavenumber points in between. All the spectra were first normalized and used as input for the ML model.

S1.2: Model builder for GAN

The GAN model employed in this study consisted of a generator network built upon the U-Net architecture with eight blocks, each comprising convolutional layers, instance normalization layers, LeakyReLU layers, and tanh activation functions.[1] Low-resolution spectra collected with an integration time of 15 s using an LR spectrometer were used as the input data for the generator. The generator network consists of multiple convolutional layers designed to capture the most valuable features and spatial hierarchies in the input data. It is a combination of an encoder and a decoder structure containing skip connection layers. The generator is crucial in transforming low-resolution Raman spectra into their high-resolution counterparts.[2] Since the generator is a U-Net-based design, it first down-samples all the information from the low-resolution Raman spectra to lower dimensional space through convolutional layers. Once the network extracts these valuable features, it starts upscaling and rebuilds the fully resolved high-resolution Raman spectra using extracted features. The discriminator is a five-block network and is composed of convolutional layers aimed at discerning between the real and the generated data. It also employs layers of convolutional, batch normalization, and Leaky ReLU. All convolutional layers utilize 4×1 spatial filters applied with a stride of size 2. The output layer of the generator and discriminator has 1×1 spatial filters applied with a stride of size 1, followed by the Tanh activation function for the generator and the sigmoid activation function for the discriminator. While the core structure of the GAN follows conventional GAN architectures, our approach lies in the specific application to Raman spectral data. We have optimized the network design for spectral resolution enhancement, tailoring convolutional layers to capture the unique features of low-resolution Raman spectra and optimized training parameters to ensure rapid convergence and high-resolution spectral generation.

Training parameters such as the learning rate, batch size, and epochs were carefully selected to optimize model performance. The model is trained over multiple epochs and tries to optimize the losses. The adversarial training process involved a competitive interplay between the generator and discriminator, with the former generating spectra to deceive the latter and the discriminator evolving to discern true high-resolution spectra. The rigorous evaluation

included generalization of performance on unseen spectra, quantitative metrics like SNR, spectral resolution, and peak intensity, and comparative analyses against spectra obtained directly from a high-resolution Raman spectrophotometer.

To train the GAN model, we employed a supervised learning strategy by partitioning the complete dataset of 2400 Raman spectra into three distinct subsets. Specifically, 70% of the data (1680 spectra) was allocated for training, enabling the model to learn the mapping between low-resolution and high-resolution spectral domains. To optimize the learning process and prevent overfitting, 15% of the data (360 spectra) was reserved as a validation set and used to monitor model performance during training iterations. Once training was complete, the remaining 15% (360 spectra) was utilized as an independent test set to evaluate the generalization capability of the model on previously unseen spectral inputs. During the training process, the model tries to adjust its weight and biases in the layer, as well as these losses from both the generator and the discriminator. The loss function, L_G from the generator network, measures how significantly the generator can deceive the discriminator network. The generator aims to minimize this loss (L_G) during the training process to effectively generate outputs indistinguishable from real high-resolution spectra. The loss function, L_D, from the discriminator network measures the discriminator's ability to classify between real and generated spectra. The discriminator aims to maximize this loss (L_D), improving its capability to differentiate between real and generated data. The parameters G and D in the loss functions represent the weights and biases, which are iteratively updated during training to improve the efficiency accuracy for the generator and discriminator network, respectively. The interplay between L_G and L_D creates a dynamic training process where the generator and discriminator compete to continuously improve in a zero-sum game framework.

The loss functions in the GAN model come from both generator (L_G) and discriminator (L_D) and were defined as[2]

$$L_G = \frac{1}{N}\sum_{n=1}^{N}(D(G(s^n), s^n) - 1)^2 + \lambda\left|\left|G(s^n) - x^n\right|\right|_1$$

$$L_D = \frac{1}{N}\sum_{n=1}^{N}(D(x^n, s^n) - 1)^2 + \left(D(G(s^n), s^n)\right)^2$$

Where a high-resolution spectrum is denoted as x^n and a corresponding low-resolution spectrum is denoted as s^n, N is the number of training spectra. The hyperparameter λ was introduced to control the magnitude of the L1 norm, a metric used to quantify the dissimilarity between vectors.[2] L1 norm was chosen to improve the SNR of the processed spectra. To train our model, we employed the Adam optimization algorithm over 177 epochs, with a learning rate of 0.0001 and an effective batch size of one.[3] These parameters were selected to optimize the model's performance in reconstructing high-resolution spectra from their low-resolution counterparts.

Due to the architectural complexity of the proposed GAN model comprising a deep convolutional generator and discriminator with a large number of trainable parameters, the training process is computationally intensive. The model was trained over approximately 164 hours on a high-performance computing (HPC) cluster utilizing two parallel GPU nodes, each equipped with NVIDIA A100 GPUs (40 GB VRAM) and 2× Intel Xeon Platinum 8358 CPUs (32 cores, 2.6 GHz, 8 CPUs). However, once the training phase is completed, the model demonstrates high inference efficiency. Specifically, it requires only ~88 milliseconds to generate a high-resolution Raman spectrum from a single low-resolution input of array size 862×1. Furthermore, the inference time scales linearly with the input array size, making the model well-suited for real-time or high-throughput spectral enhancement applications.

S1.3: Classification of GAN-generated spectra using Artificial Neural Network (ANN)

We employed an ANN architecture to classify the generated data and tuned the hyperparameters such as layer size, number of hidden neurons in each layer, and training function. The best classification model after optimization of the parameters is the ANN comprised of four layers: an input layer, two hidden layers with 17 and 18 neurons, respectively, and an output classification layer with six nodes corresponding to the categories of the samples (**Figure S1**). The evaluation of the individual models is given in **Table S2**. The ANN model in this study utilized the Bayesian regularization backpropagation algorithm for training, which iteratively updates the weights and biases based on the Levenberg-Marquardt optimization.[4] Specifically, we employed the 'trainbr' function and assessed network performance using cross-entropy error and misclassification error metrics.[5,6]

Cross-entropy loss, also known as log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. The cross-entropy loss increases as the predicted probability diverges from the actual label. In essence, it quantifies the difference between two probability distributions—the true labels and the predicted probabilities. For single-label classification, the cross-entropy loss is calculated as:

$$loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} T_{ic} ln Y_{ic}$$

Where N is the number of samples, T_{ic} is the binary indicator (0 or 1) that corresponds to whether the class label *c* is the correct classification for the i^{th} sample, and Y_{ic} is the predicted probability of the i^{th} sample being in class *c*. The misclassification error metric measures the proportion of incorrect predictions made by the model, i.e., it calculates the ratio of the number of incorrect predictions to the total number of predictions and is given by the following equation:

$$E = \frac{1}{N}\sum_{i=1}^{N} I\ (\hat{y}_i \neq y_i)$$

Where *N* is the total number of samples, $\hat{y}_i$ is the predicted label, and y_i is the true label for the i^{th} sample.

During the training state of the ANN model, we set the minimum performance gradient to 1 x e-6 and the maximum validation failure to 6. This means that if the performance of the model does not change by a margin of 1 x e-6 for six successive epochs, training will stop immediately. Upon satisfying these convergence criteria, the model is considered fully trained and ready for testing and validation. To ensure the reliability of our classification model, we conducted a 3-fold validation process by randomly partitioning the dataset into training, testing, and validation subsets. The input data was normalized, and no additional preprocessing steps were applied to promote the generalization of the model. Since we have a substantial dataset, we utilized the entire Raman spectrum without augmentation for both training and testing. High-resolution data was used to train the model. Once the model was trained, it was further tested on the generated high-resolution dataset to assess its performance. The model optimization was based on minimizing mean square error (MSE) and maximizing the coefficient of correlation (R). Convergence criteria for the model were determined by monitoring changes in MSE and R over successive iterations, with the training concluding once a predefined threshold was reached and remained unchanged for a specified number of iterations. The results were further analyzed through visualization techniques, such as confusion matrices and receiver operating characteristic (ROC) curves. Lastly, MATLAB scripts were generated to replicate the results and enable customization of the training process, ensuring reproducibility and flexibility in future experiments.

The MSE for the classification model was calculated as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)$$

Wherein *n* is the number of total observations, X_i is the true class of the sample, and $\widehat{X}_t$ is the class of the sample predicted by the model.

Although classifiers such as Support Vector Machine (SVM), k-nearest Neighbours (k-NN), Random Forest (RF), and Gradient Boosting Machines (GBM) can be used, the ANN method was opted over others because of the complex non-linear relationships inherent in Raman spectra, which might be challenging for linear models. ANNs can automatically learn features from raw data, reducing the need for manual feature extraction.[7] Beyond classification accuracy, efforts were made for post-training interpretability, employing feature analysis and visualization techniques to gain insights into the distinctive spectral attributes defining different sample categories. For the visualization of different categories of samples, we first used Principal Component Analysis (PCA) to reduce the data from higher to lower dimensions. After that, the scores obtained from the PCA were directly incorporated into the Radial Visualization (RadViz) in Orange software.

S1.4: Identification of the generated spectra of organic compounds via spectral barcoding

To enable compound classification and automated identification, we adopted the concept of Raman barcoding for the analyte molecules. A comprehensive Raman spectral barcode library was initially constructed using the experimentally obtained Raman spectra from the HR spectrometer. The high-resolution Raman spectra obtained were processed in such a way that all information regarding peak position and full-width half maxima (FWHM) was integrated into the form of barcodes. The thickness and position of vertical lines are directly related to the spectral peak position and FWHM, respectively. An in-house MATLAB script was developed to preprocess the high-resolution spectra into their respective barcodes. Initially, spectral normalization was performed within the intensity range of 0 to 1. Subsequently, a peak searching algorithm identifies peaks in the Raman spectrum based on peak prominence using the "findpeaks" function. Raman barcodes were employed to verify the identity of the

unknown sample through barcode comparison. We opted for the Structural Similarity Index (SSIM) method to measure the similarity between two barcodes. The significance of SSIM lies in its ability to provide a more accurate and human-perceptual measure of image similarity, making it particularly useful in fields such as image processing, computer vision, and remote sensing. SSIM provides a robust and perceptually relevant measure of image similarity, leveraging the structural information within images to offer a meaningful comparison metric. SSIM compares two images, I and I', by considering three components: luminance (l), contrast (c), and structure (s). The SSIM index is calculated using the following formula:[8]

$$SSIM(I, I') = [l(I, I')]^{\alpha} \cdot [c(I, I')]^{\beta} \cdot [s(I, I')]^{\gamma}$$

If a significant portion of the barcode signature of the unknown sample matched with the barcode spectrum from the library, i.e., X, then the unknown sample was identified as X. The threshold percentage of the unknown sample barcode spectrum required for identification is referred to as the percentage match criterion. Subsequently, low-resolution Raman spectra were fed into a GAN model to generate high-resolution spectral outputs. These generated spectra were then encoded into Raman barcodes and compared against the spectral library to determine the similarity, quantified as a percentage match between the unknown sample's Raman barcode and those in the library. Upon scanning the entire library, the model identified the top-scoring profiles, thereby providing the identity of the unknown sample.

If a significant portion of the barcode signature of the unknown sample matched with the barcode spectrum from the library, i.e., X, then the unknown sample was identified as X. The threshold percentage of the unknown sample barcode spectrum required for identification is referred to as the percentage match criterion. Subsequently, low-resolution Raman spectra were fed into a GAN model to generate high-resolution spectral outputs. These generated spectra were then encoded into Raman barcodes and compared against the spectral library to determine the similarity, quantified as a percentage match between the unknown sample's

Raman barcode and those in the library. Upon scanning the entire library, the model identified the top-scoring profiles, thereby providing the identity of the unknown sample.

S1.5: Calculating of Signal-to-Noise Ratio

The SNR for the low, high, and generated high-resolution Raman spectra were calculated by the formula below:[9]

$$SNR = \frac{Raman\ Peak\ Intensity}{Standard\ deviation\ of\ background\ noise}$$

$$SNR\ (dB) = 10\ log_{10}(SNR)$$

S1.6: Process of making Raman spectral barcode

For the barcoding process, we employed the "findpeaks" function in MATLAB, utilizing a peak prominence threshold set at 0.3, indicating that peaks with an SNR of 0.3 or higher are considered valid peaks within the Raman spectrum, discernible from the background noise. Once all peaks within the Raman spectrum and their FWHM values were identified and labeled, an in-house MATLAB algorithm was developed to fetch this information and integrate it into constructing the Raman barcode. Since Raman is a very sensitive technique for the detection of a sample, and every sample has a distinct Raman fingerprint region, the Raman barcode will be a unique identity that is assigned to the sample on the basis of Raman spectra. Consequently, each barcode encapsulates specific information unique to each sample, facilitating streamlined identification of unknown samples. We have created a spectral barcode library using high-resolution Raman spectra obtained from a high-resolution spectrometer. Given that the GAN has successfully transformed low-resolution Raman spectra into their high-resolution equivalents. The identifier then uses these generated high-resolution spectra to extract the FWHM and peak positions, converting them into Raman barcodes. These barcodes are subsequently scanned against the Raman spectral barcode library, and the resulting matrix represents the percentage similarity index. We use the built-

in MATLAB function 'SSIM' to calculate the similarity index of the barcode from an unknown sample against the spectral library.

Table S1: Raman band assignment of different samples: aspirin, ibuprofen, PCM, ranitidine, 4-MBA, and 4-NTP.[10,11,12,13,14]

Raman Shift (cm^{-1})	Raman band assignment
1030	Aromatic rings
1200	-OH group substitution
1300	C-C bond
1600	C=O stretching
700-830	γ C-H
1231	C-C ring stretching
858	Ring breathing
791	CNC ring stretching
1319	Amide III
1558	Amide II N-H in plane deformation
1607	skeletal aryl C-C ring stretching
1644	Amide I
841	out-of-plane C-H bending
1441, 1472	aryl C-C stretch
1511	aryl C-H symmetric bends
1587	NO_2 asymmetric stretching
1554, 1408	NO_2 symmetric stretching
1483, 1376	C-N symmetric stretching
1450	C-C symmetric stretching
1306	C-H in-plane bending
1263, 1248	C-H out of plane bending

Table S2: The model optimization table for various parameters.

Model No.	Training Function	Hidden Neurons	Test Loss	Model No.	Training Function	Hidden Neurons	Test Loss
1	trainlm	[6]	0.025981	45	trainlm	[8, 19]	0.02637
2	trainlm	[3, 6]	0.028332	46	trainlm	[8, 20]	0.027906
3	trainlm	[4, 6]	0.026843	47	trainlm	[9, 10]	0.021962
4	trainlm	[4, 10]	0.029288	48	trainlm	[9, 11]	0.021414
5	trainlm	[4, 11]	0.029144	49	trainlm	[9, 13]	0.023985
6	trainlm	[4, 12]	0.023078	50	trainlm	[9, 15]	0.023128
7	trainlm	[4, 15]	0.029084	51	trainlm	[9, 16]	0.024728
8	trainlm	[5, 10]	0.026997	52	trainlm	[9, 17]	0.027728
9	trainlm	[5, 11]	0.024478	53	trainlm	[9, 18]	0.023069
10	trainlm	[5, 12]	0.023113	54	trainlm	[9, 19]	0.022849
11	trainlm	[5, 13]	0.02859	55	trainlm	[10, 14]	0.029717
12	trainlm	[5, 14]	0.027967	56	trainlm	[10, 15]	0.026676
13	trainlm	[5, 16]	0.029415	57	trainlm	[10, 17]	0.029091
14	trainlm	[5, 18]	0.024838	58	trainlm	[10, 20]	0.01982
15	trainlm	[5, 19]	0.027732	59	trainlm	[11, 14]	0.023211
16	trainlm	[5, 20]	0.028819	60	trainlm	[11, 15]	0.025901
17	trainlm	[6, 7]	0.02589	61	trainlm	[11, 16]	0.022485
18	trainlm	[6, 9]	0.025331	62	trainlm	[11, 17]	0.024967
19	trainlm	[6, 10]	0.022424	63	trainlm	[11, 18]	0.027611
20	trainlm	[6, 11]	0.022461	64	trainlm	[11, 19]	0.025603
21	trainlm	[6, 12]	0.022132	65	trainlm	[11, 20]	0.027897
22	trainlm	[6, 13]	0.02589	66	trainlm	[12, 14]	0.022319
23	trainlm	[6, 14]	0.02808	67	trainlm	[12, 16]	0.019906
24	trainlm	[6, 15]	0.021693	68	trainlm	[12, 17]	0.026521
25	trainlm	[6, 16]	0.021657	69	trainlm	[12, 18]	0.022467
26	trainlm	[6, 17]	0.024102	70	trainlm	[12, 19]	0.022086
27	trainlm	[6, 18]	0.028086	71	trainlm	[13, 16]	0.027623
28	trainlm	[6, 19]	0.02007	72	trainlm	[13, 17]	0.024226
29	trainlm	[6, 20]	0.018801	73	trainlm	[13, 20]	0.029899
30	trainlm	[7, 9]	0.024369	74	trainlm	[14, 15]	0.029871
31	trainlm	[7, 10]	0.022259	75	trainlm	[14, 16]	0.018615
32	trainlm	[7, 11]	0.027534	76	trainlm	[14, 17]	0.026844
33	trainlm	[7, 12]	0.023687	77	trainlm	[15, 16]	0.026204
34	trainlm	[7, 13]	0.024874	78	trainlm	[15, 17]	0.025169
35	trainlm	[7, 14]	0.024143	79	trainlm	[16, 18]	0.019279
36	trainlm	[7, 16]	0.025499	80	trainlm	[16, 20]	0.026439
37	trainlm	[7, 18]	0.02622	81	trainlm	[17, 18]	0.029288
38	trainlm	[7, 19]	0.022173	82	trainlm	[17, 20]	0.022176
39	trainlm	[8, 9]	0.024886	83	trainlm	[18, 19]	0.027043
40	trainlm	[8, 10]	0.019097	84	trainlm	[3, 4, 7]	0.024825
41	trainlm	[8, 13]	0.025541	85	trainlm	[3, 4, 20]	0.028591
42	trainlm	[8, 14]	0.026174	86	trainlm	[3, 5, 16]	0.029862
43	trainlm	[8, 15]	0.028847	87	trainlm	[3, 6, 17]	0.024444
44	trainlm	[8, 17]	0.028759	88	trainlm	[3, 7, 12]	0.027207

Model No.	Training Function	Hidden Neurons	Test Loss	Model No.	Training Function	Hidden Neurons	Test Loss
89	trainlm	[3, 10, 12]	0.023092	133	trainlm	[4, 12, 13]	0.023064
90	trainlm	[3, 11, 15]	0.028372	134	trainlm	[4, 12, 14]	0.021336
91	trainlm	[3, 11, 18]	0.028773	135	trainlm	[4, 12, 15]	0.022402
92	trainlm	[3, 12, 17]	0.029499	136	trainlm	[4, 12, 16]	0.023035
93	trainlm	[3, 13, 20]	0.02803	137	trainlm	[4, 12, 17]	0.022225
94	trainlm	[3, 14, 16]	0.029983	138	trainlm	[4, 12, 18]	0.027938
95	trainlm	[3, 14, 19]	0.026622	139	trainlm	[4, 12, 19]	0.027287
96	trainlm	[3, 15, 16]	0.028687	140	trainlm	[4, 12, 20]	0.022275
97	trainlm	[3, 16, 17]	0.02784	141	trainlm	[4, 13, 14]	0.02507
98	trainlm	[3, 18, 20]	0.027252	142	trainlm	[4, 13, 20]	0.021553
99	trainlm	[3, 19, 20]	0.026676	143	trainlm	[4, 14, 16]	0.026146
100	trainlm	[4, 5, 9]	0.027233	144	trainlm	[4, 14, 17]	0.024048
101	trainlm	[4, 5, 15]	0.022009	145	trainlm	[4, 14, 18]	0.021127
102	trainlm	[4, 5, 17]	0.021853	146	trainlm	[4, 14, 20]	0.023905
103	trainlm	[4, 5, 20]	0.021913	147	trainlm	[4, 15, 17]	0.023299
104	trainlm	[4, 6, 7]	0.028072	148	trainlm	[4, 15, 18]	0.026942
105	trainlm	[4, 6, 11]	0.029618	149	trainlm	[4, 15, 19]	0.025296
106	trainlm	[4, 6, 12]	0.024806	150	trainlm	[4, 15, 20]	0.026287
107	trainlm	[4, 6, 16]	0.028125	151	trainlm	[4, 16, 17]	0.028423
108	trainlm	[4, 6, 18]	0.027009	152	trainlm	[4, 16, 18]	0.022115
109	trainlm	[4, 7, 8]	0.024657	153	trainlm	[4, 16, 19]	0.020908
110	trainlm	[4, 7, 9]	0.02925	154	trainlm	[4, 16, 20]	0.021028
111	trainlm	[4, 7, 10]	0.027788	155	trainlm	[4, 17, 20]	0.024154
112	trainlm	[4, 7, 11]	0.026738	156	trainlm	[4, 18, 19]	0.019465
113	trainlm	[4, 7, 13]	0.028909	157	trainlm	[4, 19, 20]	0.020579
114	trainlm	[4, 7, 18]	0.023425	158	trainlm	[5, 6, 8]	0.026558
115	trainlm	[4, 8, 11]	0.022378	159	trainlm	[5, 6, 11]	0.028853
116	trainlm	[4, 8, 12]	0.025057	160	trainlm	[5, 6, 12]	0.019281
117	trainlm	[4, 8, 14]	0.028675	161	trainlm	[5, 6, 13]	0.021773
118	trainlm	[4, 8, 15]	0.027409	162	trainlm	[5, 6, 18]	0.024057
119	trainlm	[4, 8, 18]	0.025703	163	trainlm	[5, 7, 8]	0.029874
120	trainlm	[4, 9, 11]	0.024389	164	trainlm	[5, 7, 9]	0.026643
121	trainlm	[4, 9, 12]	0.029705	165	trainlm	[5, 7, 12]	0.019883
122	trainlm	[4, 9, 14]	0.02381	166	trainlm	[5, 7, 14]	0.027047
123	trainlm	[4, 9, 15]	0.023733	167	trainlm	[5, 7, 15]	0.025317
124	trainlm	[4, 9, 16]	0.026554	168	trainlm	[5, 7, 16]	0.025411
125	trainlm	[4, 9, 18]	0.025992	169	trainlm	[5, 7, 17]	0.029508
126	trainlm	[4, 9, 19]	0.025909	170	trainlm	[5, 7, 18]	0.028147
127	trainlm	[4, 10, 15]	0.019944	171	trainlm	[5, 7, 20]	0.02521
128	trainlm	[4, 10, 20]	0.028791	172	trainlm	[5, 8, 9]	0.01958
129	trainlm	[4, 11, 13]	0.027324	173	trainlm	[5, 8, 11]	0.025317
130	trainlm	[4, 11, 16]	0.02285	174	trainlm	[5, 8, 14]	0.028328
131	trainlm	[4, 11, 17]	0.02681	175	trainlm	[5, 8, 16]	0.025291
132	trainlm	[4, 11, 20]	0.024606	176	trainlm	[5, 8, 17]	0.028868

Model No.	Training Function	Hidden Neurons	Test Loss	Model No.	Training Function	Hidden Neurons	Test Loss
177	trainlm	[5, 8, 18]	0.024276	221	trainlm	[5, 17, 19]	0.022788
178	trainlm	[5, 8, 19]	0.024267	222	trainlm	[5, 17, 20]	0.025473
179	trainlm	[5, 8, 20]	0.026215	223	trainlm	[5, 18, 19]	0.029239
180	trainlm	[5, 9, 10]	0.023812	224	trainlm	[6, 7, 9]	0.02589
181	trainlm	[5, 9, 13]	0.027888	225	trainlm	[6, 7, 11]	0.019419
182	trainlm	[5, 9, 15]	0.026313	226	trainlm	[6, 7, 12]	0.025502
183	trainlm	[5, 9, 17]	0.020138	227	trainlm	[6, 7, 13]	0.019197
184	trainlm	[5, 9, 19]	0.022191	228	trainlm	[6, 7, 14]	0.028957
185	trainlm	[5, 9, 20]	0.026173	229	trainlm	[6, 7, 15]	0.02706
186	trainlm	[5, 10, 11]	0.028822	230	trainlm	[6, 7, 16]	0.027207
187	trainlm	[5, 10, 13]	0.022384	231	trainlm	[6, 7, 20]	0.023248
188	trainlm	[5, 10, 14]	0.029394	232	trainlm	[6, 8, 13]	0.027517
189	trainlm	[5, 10, 16]	0.027266	233	trainlm	[6, 8, 14]	0.022826
190	trainlm	[5, 10, 17]	0.02577	234	trainlm	[6, 8, 15]	0.025366
191	trainlm	[5, 10, 18]	0.026393	235	trainlm	[6, 8, 16]	0.028407
192	trainlm	[5, 10, 19]	0.025652	236	trainlm	[6, 8, 17]	0.026367
193	trainlm	[5, 10, 20]	0.026157	237	trainlm	[6, 8, 20]	0.022482
194	trainlm	[5, 11, 12]	0.02386	238	trainlm	[6, 9, 10]	0.020926
195	trainlm	[5, 11, 14]	0.022098	239	trainlm	[6, 9, 11]	0.020374
196	trainlm	[5, 11, 15]	0.029555	240	trainlm	[6, 9, 12]	0.022418
197	trainlm	[5, 11, 16]	0.020461	241	trainlm	[6, 9, 13]	0.021767
198	trainlm	[5, 11, 17]	0.020628	242	trainlm	[6, 9, 14]	0.026815
199	trainlm	[5, 11, 19]	0.024751	243	trainlm	[6, 9, 15]	0.028121
200	trainlm	[5, 12, 14]	0.025937	244	trainlm	[6, 9, 17]	0.022267
201	trainlm	[5, 12, 15]	0.01989	245	trainlm	[6, 9, 18]	0.024894
202	trainlm	[5, 12, 18]	0.026095	246	trainlm	[6, 9, 19]	0.028824
203	trainlm	[5, 12, 19]	0.028442	247	trainlm	[6, 9, 20]	0.021732
204	trainlm	[5, 12, 20]	0.020484	248	trainlm	[6, 10, 11]	0.028234
205	trainlm	[5, 13, 16]	0.028746	249	trainlm	[6, 10, 12]	0.019828
206	trainlm	[5, 13, 17]	0.025889	250	trainlm	[6, 10, 14]	0.025194
207	trainlm	[5, 13, 19]	0.025197	251	trainlm	[6, 10, 17]	0.025453
208	trainlm	[5, 14, 15]	0.022911	252	trainlm	[6, 10, 18]	0.025026
209	trainlm	[5, 14, 16]	0.025035	253	trainlm	[6, 10, 20]	0.020242
210	trainlm	[5, 14, 17]	0.023382	254	trainlm	[6, 11, 12]	0.027642
211	trainlm	[5, 14, 18]	0.023999	255	trainlm	[6, 11, 14]	0.02275
212	trainlm	[5, 14, 19]	0.023317	256	trainlm	[6, 11, 15]	0.024734
213	trainlm	[5, 14, 20]	0.027769	257	trainlm	[6, 11, 16]	0.026935
214	trainlm	[5, 15, 16]	0.025336	258	trainlm	[6, 11, 17]	0.026464
215	trainlm	[5, 15, 17]	0.017589	259	trainlm	[6, 11, 19]	0.025969
216	trainlm	[5, 15, 18]	0.029869	260	trainlm	[6, 12, 13]	0.020462
217	trainlm	[5, 15, 19]	0.021986	261	trainlm	[6, 12, 15]	0.024411
218	trainlm	[5, 16, 17]	0.026385	262	trainlm	[6, 12, 16]	0.020253
219	trainlm	[5, 16, 19]	0.025829	263	trainlm	[6, 12, 17]	0.027938
220	trainlm	[5, 16, 20]	0.020876	264	trainlm	[6, 12, 19]	0.029521

Model No.	Training Function	Hidden Neurons	Test Loss	Model No.	Training Function	Hidden Neurons	Test Loss
265	trainlm	[6, 12, 20]	0.01787	309	trainlm	[7, 11, 14]	0.020266
266	trainlm	[6, 13, 14]	0.021036	310	trainlm	[7, 11, 15]	0.021781
267	trainlm	[6, 13, 17]	0.02246	311	trainlm	[7, 11, 16]	0.022872
268	trainlm	[6, 13, 18]	0.020947	312	trainlm	[7, 11, 17]	0.022325
269	trainlm	[6, 13, 19]	0.022918	313	trainlm	[7, 11, 18]	0.017691
270	trainlm	[6, 13, 20]	0.023887	314	trainlm	[7, 11, 20]	0.024701
271	trainlm	[6, 14, 15]	0.016728	315	trainlm	[7, 12, 13]	0.022607
272	trainlm	[6, 14, 17]	0.019388	316	trainlm	[7, 12, 14]	0.025172
273	trainlm	[6, 14, 18]	0.028385	317	trainlm	[7, 12, 17]	0.021569
274	trainlm	[6, 15, 16]	0.029194	318	trainlm	[7, 12, 18]	0.021456
275	trainlm	[6, 15, 17]	0.028094	319	trainlm	[7, 12, 19]	0.020822
276	trainlm	[6, 15, 18]	0.018153	320	trainlm	[7, 12, 20]	0.027649
277	trainlm	[6, 15, 19]	0.017323	321	trainlm	[7, 13, 15]	0.02543
278	trainlm	[6, 16, 18]	0.021136	322	trainlm	[7, 13, 16]	0.025565
279	trainlm	[6, 16, 19]	0.019058	323	trainlm	[7, 13, 17]	0.021566
280	trainlm	[6, 16, 20]	0.027041	324	trainlm	[7, 13, 19]	0.029497
281	trainlm	[6, 17, 18]	0.02074	325	trainlm	[7, 13, 20]	0.017737
282	trainlm	[6, 17, 19]	0.023886	326	trainlm	[7, 14, 15]	0.029436
283	trainlm	[6, 17, 20]	0.024397	327	trainlm	[7, 14, 17]	0.020334
284	trainlm	[6, 19, 20]	0.021832	328	trainlm	[7, 14, 19]	0.024916
285	trainlm	[7, 8, 9]	0.025043	329	trainlm	[7, 14, 20]	0.024732
286	trainlm	[7, 8, 10]	0.02491	330	trainlm	[7, 15, 16]	0.023846
287	trainlm	[7, 8, 11]	0.028189	331	trainlm	[7, 15, 17]	0.019015
288	trainlm	[7, 8, 12]	0.022613	332	trainlm	[7, 15, 18]	0.028382
289	trainlm	[7, 8, 15]	0.02604	333	trainlm	[7, 15, 19]	0.027256
290	trainlm	[7, 8, 16]	0.029233	334	trainlm	[7, 15, 20]	0.018687
291	trainlm	[7, 8, 17]	0.023669	335	trainlm	[7, 16, 17]	0.020564
292	trainlm	[7, 9, 10]	0.025732	336	trainlm	[7, 16, 18]	0.02146
293	trainlm	[7, 9, 14]	0.026594	337	trainlm	[7, 16, 19]	0.019281
294	trainlm	[7, 9, 15]	0.027803	338	trainlm	[7, 16, 20]	0.025033
295	trainlm	[7, 9, 16]	0.018886	339	trainlm	[7, 17, 18]	0.027956
296	trainlm	[7, 9, 18]	0.024837	340	trainlm	[7, 17, 19]	0.022673
297	trainlm	[7, 9, 19]	0.021873	341	trainlm	[7, 17, 20]	0.022018
298	trainlm	[7, 9, 20]	0.020977	342	trainlm	[7, 18, 19]	0.021809
299	trainlm	[7, 10, 11]	0.018921	343	trainlm	[7, 18, 20]	0.024642
300	trainlm	[7, 10, 12]	0.019845	344	trainlm	[8, 9, 10]	0.020977
301	trainlm	[7, 10, 13]	0.029102	345	trainlm	[8, 9, 13]	0.017213
302	trainlm	[7, 10, 14]	0.026593	346	trainlm	[8, 9, 15]	0.026605
303	trainlm	[7, 10, 16]	0.025991	347	trainlm	[8, 9, 16]	0.029001
304	trainlm	[7, 10, 17]	0.025178	348	trainlm	[8, 9, 17]	0.020083
305	trainlm	[7, 10, 18]	0.018749	349	trainlm	[8, 9, 18]	0.01756
306	trainlm	[7, 10, 19]	0.022861	350	trainlm	[8, 9, 19]	0.016675
307	trainlm	[7, 10, 20]	0.021184	351	trainlm	[8, 10, 11]	0.021337
308	trainlm	[7, 11, 13]	0.028366	352	trainlm	[8, 10, 13]	0.024706

Model No.	Training Function	Hidden Neurons	Test Loss	Model No.	Training Function	Hidden Neurons	Test Loss
353	trainlm	[8, 10, 14]	0.022795	397	trainlm	[9, 10, 13]	0.029775
354	trainlm	[8, 10, 15]	0.025777	398	trainlm	[9, 10, 14]	0.02176
355	trainlm	[8, 10, 16]	0.024821	399	trainlm	[9, 10, 15]	0.018072
356	trainlm	[8, 10, 17]	0.025039	400	trainlm	[9, 10, 16]	0.027963
357	trainlm	[8, 10, 18]	0.021865	401	trainlm	[9, 10, 17]	0.027865
358	trainlm	[8, 10, 19]	0.023573	402	trainlm	[9, 10, 18]	0.020186
359	trainlm	[8, 10, 20]	0.02559	403	trainlm	[9, 10, 19]	0.02074
360	trainlm	[8, 11, 12]	0.023049	404	trainlm	[9, 10, 20]	0.021299
361	trainlm	[8, 11, 14]	0.017586	405	trainlm	[9, 11, 12]	0.029742
362	trainlm	[8, 11, 15]	0.023202	406	trainlm	[9, 11, 13]	0.021462
363	trainlm	[8, 11, 16]	0.023724	407	trainlm	[9, 11, 14]	0.023381
364	trainlm	[8, 11, 17]	0.025552	408	trainlm	[9, 11, 15]	0.020893
365	trainlm	[8, 11, 18]	0.02159	409	trainlm	[9, 11, 16]	0.027505
366	trainlm	[8, 11, 19]	0.018616	410	trainlm	[9, 11, 19]	0.017625
367	trainlm	[8, 12, 13]	0.022627	411	trainlm	[9, 12, 14]	0.029556
368	trainlm	[8, 12, 14]	0.022852	412	trainlm	[9, 12, 15]	0.019361
369	trainlm	[8, 12, 16]	0.016874	413	trainlm	[9, 12, 18]	0.02014
370	trainlm	[8, 12, 17]	0.023433	414	trainlm	[9, 12, 19]	0.016432
371	trainlm	[8, 12, 18]	0.026073	415	trainlm	[9, 12, 20]	0.020093
372	trainlm	[8, 12, 20]	0.021329	416	trainlm	[9, 13, 14]	0.021387
373	trainlm	[8, 13, 14]	0.028235	417	trainlm	[9, 13, 15]	0.024709
374	trainlm	[8, 13, 15]	0.020092	418	trainlm	[9, 13, 16]	0.022693
375	trainlm	[8, 13, 16]	0.024352	419	trainlm	[9, 13, 17]	0.023471
376	trainlm	[8, 13, 17]	0.025584	420	trainlm	[9, 13, 18]	0.018945
377	trainlm	[8, 13, 18]	0.027836	421	trainlm	[9, 13, 19]	0.018725
378	trainlm	[8, 13, 19]	0.024526	422	trainlm	[9, 13, 20]	0.020318
379	trainlm	[8, 13, 20]	0.017419	423	trainlm	[9, 14, 15]	0.025842
380	trainlm	[8, 14, 15]	0.023296	424	trainlm	[9, 14, 16]	0.025815
381	trainlm	[8, 14, 16]	0.017873	425	trainlm	[9, 14, 17]	0.020473
382	trainlm	[8, 14, 18]	0.018982	426	trainlm	[9, 14, 18]	0.020376
383	trainlm	[8, 14, 19]	0.027825	427	trainlm	[9, 14, 20]	0.018514
384	trainlm	[8, 14, 20]	0.02962	428	trainlm	[9, 15, 18]	0.02099
385	trainlm	[8, 15, 16]	0.02393	429	trainlm	[9, 15, 19]	0.025796
386	trainlm	[8, 15, 17]	0.023455	430	trainlm	[9, 16, 17]	0.017569
387	trainlm	[8, 15, 18]	0.028146	431	trainlm	[9, 16, 18]	0.023721
388	trainlm	[8, 15, 20]	0.018414	432	trainlm	[9, 16, 19]	0.015612
389	trainlm	[8, 16, 17]	0.017161	433	trainlm	[9, 16, 20]	0.017613
390	trainlm	[8, 16, 18]	0.019528	434	trainlm	[9, 17, 18]	0.017676
391	trainlm	[8, 16, 19]	0.0189	435	trainlm	[9, 17, 20]	0.025094
392	trainlm	[8, 17, 19]	0.020321	436	trainlm	[9, 18, 19]	0.019654
393	trainlm	[8, 18, 19]	0.025698	437	trainlm	[9, 18, 20]	0.026639
394	trainlm	[8, 18, 20]	0.020604	438	trainlm	[9, 19, 20]	0.021889
395	trainlm	[8, 19, 20]	0.016036	439	trainlm	[10, 11, 12]	0.023553
396	trainlm	[9, 10, 12]	0.0226	440	trainlm	[10, 11, 13]	0.025543

Model No.	Training Function	Hidden Neurons	Test Loss	Model No.	Training Function	Hidden Neurons	Test Loss
441	trainlm	[10, 11, 14]	0.029434	485	trainbr	[5, 11]	0.014153
442	trainlm	[10, 11, 15]	0.02741	486	trainbr	[5, 12]	0.014792
443	trainlm	[10, 11, 16]	0.021277	487	trainbr	[5, 13]	0.014694
444	trainlm	[10, 11, 18]	0.019751	488	trainbr	[5, 14]	0.011732
445	trainlm	[10, 11, 19]	0.023231	489	trainbr	[5, 15]	0.011909
446	trainlm	[10, 12, 13]	0.018263	490	trainbr	[5, 16]	0.014226
447	trainlm	[10, 12, 14]	0.019684	491	trainbr	[5, 18]	0.011923
448	trainlm	[10, 12, 15]	0.017852	492	trainbr	[5, 19]	0.012937
449	trainlm	[10, 12, 17]	0.019541	493	trainbr	[5, 20]	0.010302
450	trainlm	[10, 12, 18]	0.019565	494	trainbr	[6, 7]	0.012969
451	trainlm	[10, 12, 19]	0.022747	495	trainbr	[6, 8]	0.012761
452	trainlm	[10, 12, 20]	0.026328	496	trainbr	[6, 9]	0.013041
453	trainlm	[10, 13, 14]	0.02943	497	trainbr	[6, 11]	0.017278
454	trainlm	[10, 13, 15]	0.018051	498	trainbr	[6, 12]	0.010242
455	trainlm	[10, 13, 16]	0.024386	499	trainbr	[6, 14]	0.01513
456	trainlm	[10, 13, 18]	0.029167	500	trainbr	[6, 15]	0.012497
457	trainlm	[10, 13, 19]	0.022914	501	trainbr	[6, 16]	0.009478
458	trainlm	[10, 13, 20]	0.025944	502	trainbr	[6, 17]	0.016224
459	trainlm	[10, 14, 15]	0.021652	503	trainbr	[6, 18]	0.011696
460	trainlm	[10, 14, 16]	0.022042	504	trainbr	[6, 19]	0.012246
461	trainlm	[10, 14, 17]	0.020876	505	trainbr	[6, 20]	0.010943
462	trainlm	[10, 14, 18]	0.028207	506	trainbr	[7, 8]	0.008514
463	trainlm	[10, 14, 19]	0.027713	507	trainbr	[7, 9]	0.01053
464	trainlm	[10, 14, 20]	0.021265	508	trainbr	[7, 10]	0.008429
465	trainlm	[10, 15, 17]	0.017749	509	trainbr	[7, 11]	0.009379
466	trainbr	[5]	0.014153	510	trainbr	[7, 12]	0.01313
467	trainbr	[6]	0.015141	511	trainbr	[7, 13]	0.009743
468	trainbr	[3, 5]	0.014524	512	trainbr	[7, 14]	0.014942
469	trainbr	[3, 20]	0.019732	513	trainbr	[7, 15]	0.010079
470	trainbr	[4, 5]	0.01199	514	trainbr	[7, 16]	0.011815
471	trainbr	[4, 8]	0.013834	515	trainbr	[7, 17]	0.012273
472	trainbr	[4, 10]	0.018487	516	trainbr	[7, 18]	0.012879
473	trainbr	[4, 11]	0.018422	517	trainbr	[7, 19]	0.011921
474	trainbr	[4, 12]	0.015798	518	trainbr	[7, 20]	0.013182
475	trainbr	[4, 13]	0.014719	519	trainbr	[8, 9]	0.010463
476	trainbr	[4, 14]	0.015584	520	trainbr	[8, 10]	0.016625
477	trainbr	[4, 15]	0.013352	521	trainbr	[8, 11]	0.011596
478	trainbr	[4, 16]	0.015353	522	trainbr	[8, 12]	0.011678
479	trainbr	[4, 17]	0.01194	523	trainbr	[8, 13]	0.011296
480	trainbr	[4, 20]	0.014313	524	trainbr	[8, 14]	0.010791
481	trainbr	[5, 6]	0.011728	525	trainbr	[8, 15]	0.015623
482	trainbr	[5, 7]	0.012902	526	trainbr	[8, 16]	0.00825
483	trainbr	[5, 8]	0.012653	527	trainbr	[8, 17]	0.011503
484	trainbr	[5, 10]	0.011425	528	trainbr	[8, 18]	0.012841

Model No.	Training Function	Hidden Neurons	Test Loss	Model No.	Training Function	Hidden Neurons	Test Loss
529	trainbr	[8, 19]	0.008527	573	trainbr	[13, 18]	0.014217
530	trainbr	[8, 20]	0.011129	574	trainbr	[13, 19]	0.010559
531	trainbr	[9, 10]	0.011004	575	trainbr	[13, 20]	0.012732
532	trainbr	[9, 11]	0.016558	576	trainbr	[14, 15]	0.010466
533	trainbr	[9, 12]	0.017143	577	trainbr	[14, 17]	0.010628
534	trainbr	[9, 13]	0.011186	578	trainbr	[14, 18]	0.018114
535	trainbr	[9, 14]	0.010405	579	trainbr	[14, 19]	0.008564
536	trainbr	[9, 15]	0.013585	580	trainbr	[14, 20]	0.015524
537	trainbr	[9, 16]	0.010382	581	trainbr	[15, 16]	0.009957
538	trainbr	[9, 17]	0.011499	582	trainbr	[15, 17]	0.009106
539	trainbr	[9, 18]	0.009907	583	trainbr	[15, 18]	0.012153
540	trainbr	[9, 19]	0.011854	584	trainbr	[15, 19]	0.011849
541	trainbr	[9, 20]	0.011469	585	trainbr	[15, 20]	0.011703
542	trainbr	[10, 11]	0.009864	586	trainbr	[16, 17]	0.011123
543	trainbr	[10, 12]	0.013366	587	trainbr	[16, 18]	0.010621
544	trainbr	[10, 13]	0.013657	588	trainbr	[16, 19]	0.011178
545	trainbr	[10, 14]	0.013183	589	trainbr	[16, 20]	0.010883
546	trainbr	[10, 15]	0.012048	590	trainbr	[17, 18]	0.007922
547	trainbr	[10, 16]	0.011199	591	trainbr	[17, 19]	0.011424
548	trainbr	[10, 17]	0.013794	592	trainbr	[17, 20]	0.011471
549	trainbr	[10, 18]	0.014148	593	trainbr	[18, 19]	0.01041
550	trainbr	[10, 19]	0.013545	594	trainbr	[18, 20]	0.010753
551	trainbr	[10, 20]	0.009219	595	trainbr	[19, 20]	0.011261
552	trainbr	[11, 12]	0.013531				
553	trainbr	[11, 13]	0.010876				
554	trainbr	[11, 14]	0.014877				
555	trainbr	[11, 15]	0.009262				
556	trainbr	[11, 16]	0.012353				
557	trainbr	[11, 17]	0.015619				
558	trainbr	[11, 18]	0.014471				
559	trainbr	[11, 19]	0.008801				
560	trainbr	[11, 20]	0.00969				
561	trainbr	[12, 13]	0.012397				
562	trainbr	[12, 14]	0.012743				
563	trainbr	[12, 15]	0.01034				
564	trainbr	[12, 16]	0.009265				
565	trainbr	[12, 17]	0.016304				
566	trainbr	[12, 18]	0.013469				
567	trainbr	[12, 19]	0.010983				
568	trainbr	[12, 20]	0.008335				
569	trainbr	[13, 14]	0.011				
570	trainbr	[13, 15]	0.011316				
571	trainbr	[13, 16]	0.009451				
572	trainbr	[13, 17]	0.008701				

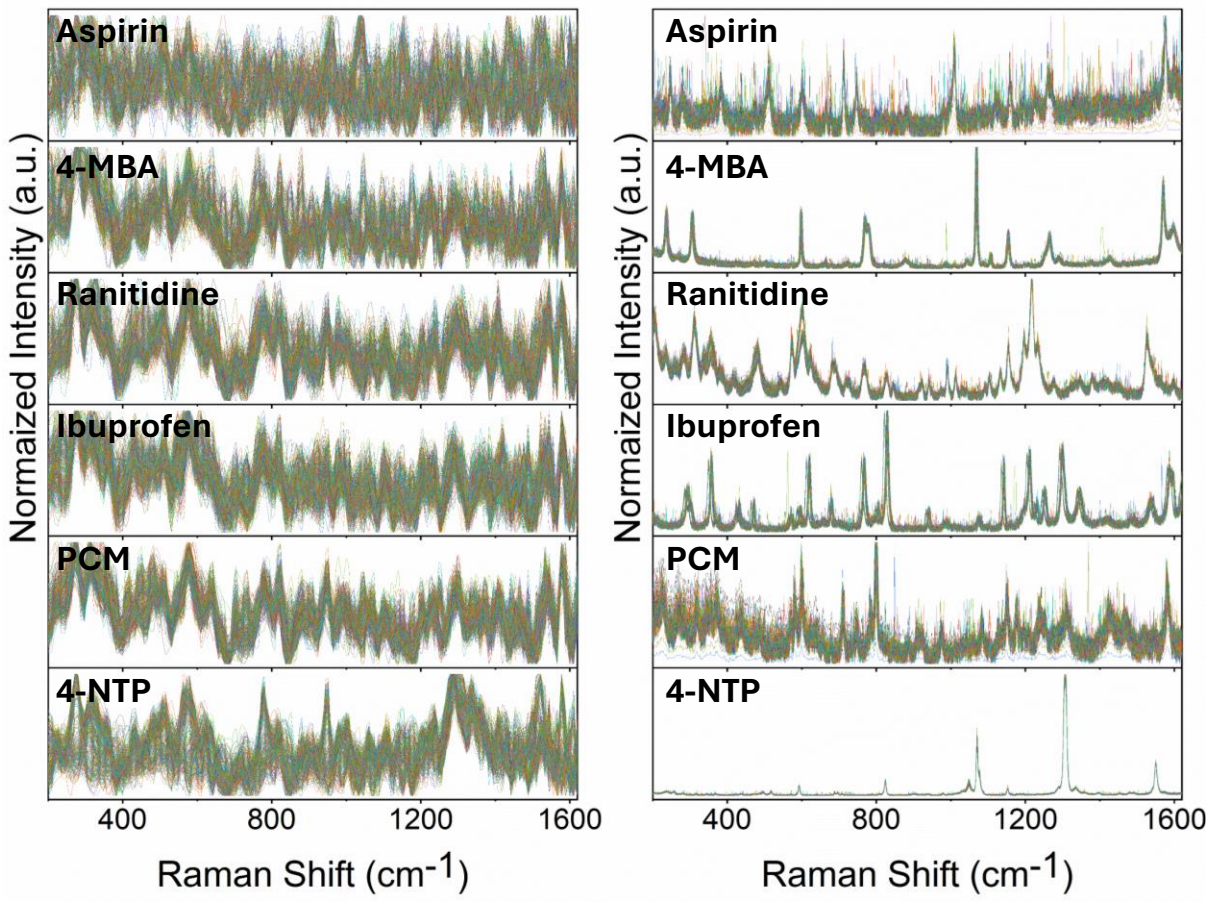

Figure S1: The low-resolution (left) and high-resolution (right) 400 Raman spectra each of different samples taken in solid form using a 785 nm laser source.

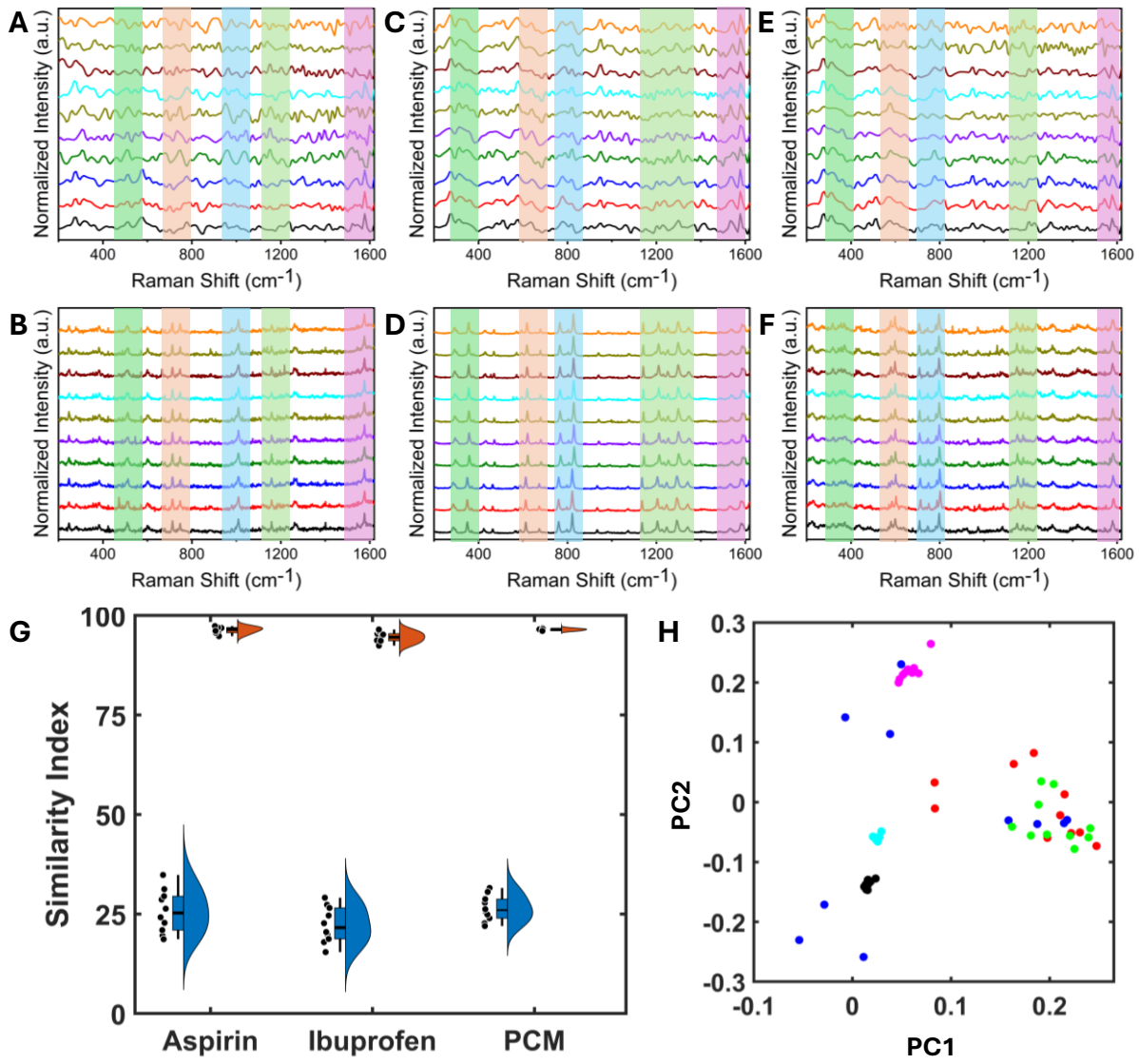

Figure S2: The additional low-resolution Raman spectra (10 spectra from each sample class) (A) Aspirin, (C) Ibuprofen, and (E) PCM, where the highlighted part in the low-resolution spectra shows the spectral differences (similarity index of 23.96%, 21.01%, and 23.80%, respectively) within the same sample under identical conditions. This additional low-resolution Raman spectra (A, C, and E) was fed to the trained GAN model to get the high-resolution instance. The highlighted part shows that instead of the spectral differences in low-resolution profile, the model capable to rebuild these lost features and generate the high-resolution counterpart (B) Aspirin, (D) Ibuprofen, and (E) PCM (similarity index of 96.70%, 94.31%, and 96.30%, respectively). (G) The violin plot shows the variation of similarity index in the additional low-resolution and GAN-generated high resolution Raman spectra shows less than 25% similarity between low-resolution profile while the GAN-generated instances show more than 90% similarity index. (H) The scatter plot from the PCA scores shows the highly random fluctuation (more spread in data points) in the low-resolution profile and minimum in the GAN-generated high-resolution Raman spectra.

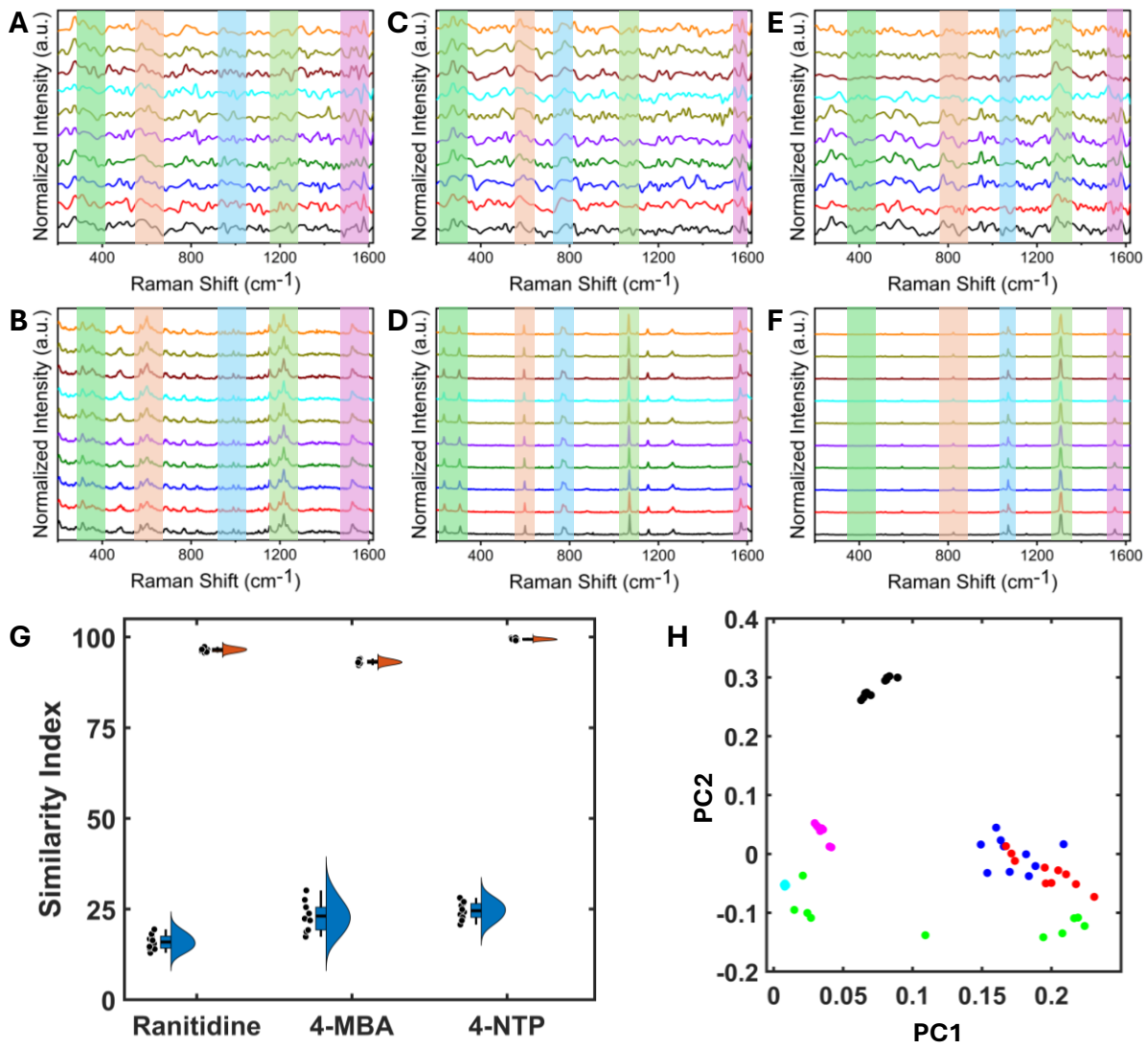

Figure S3: The additional low-resolution Raman spectra (10 spectra from each sample class) (A) Ranitidine, (C) 4-MBA, and (E) 4-NTP, where the highlighted part in the low-resolution spectra shows the spectral differences (similarity index of 15.39%, 22.21%, and 24.05%, respectively) within the same sample under identical conditions. This additional low-resolution Raman spectra (A, C, and E) was fed to the trained GAN model to get the high-resolution instance. The highlighted part shows that instead of the spectral differences in low-resolution profile, the model capable to rebuild these lost features and generate the high-resolution counterpart (B) Ranitidine, (D) 4-MBA, and (F) 4-NTP (similarity index of 96.45%, 93.13%, and 99.57%, respectively). (G) The violin plot shows the variation of similarity index in the additional low-resolution and GAN-generated high resolution Raman spectra shows less than 25% similarity between low-resolution profile while the GAN-generated instances show more than 90% similarity index. (H) The scatter plot from the PCA scores shows the highly random fluctuation (more spread in data points) in the low-resolution profile and minimum in the GAN-generated high-resolution Raman spectra.

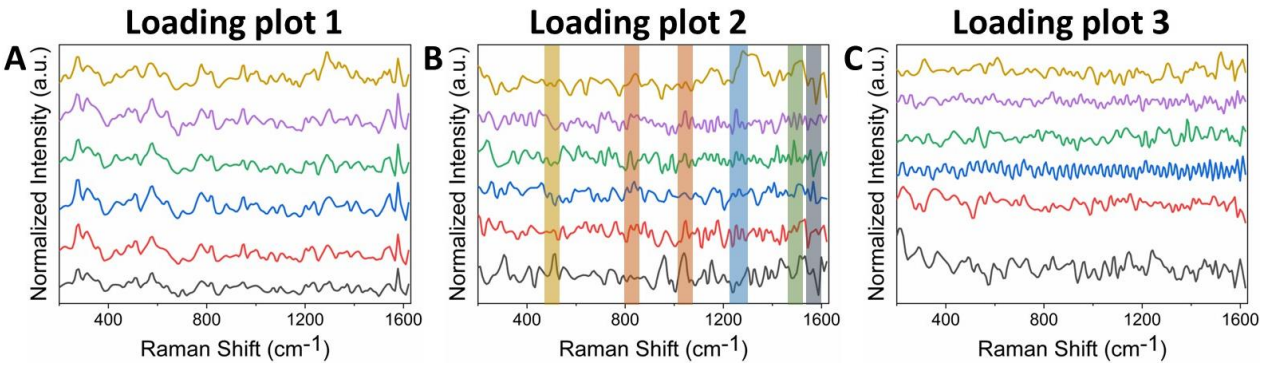

Figure S4: PCA loadings were utilized to observe the differences between the visually similar-looking spectra from the different sample classes. The Raman spectral features appeared visually identical in Loading Plot 1, but there is a major difference between the Raman features of the different sample classes as shown in Loading Plot 2. These differences, which are not visually apparent in the original low-resolution spectra, can be detected through PCA. This shows that the visually similar spectra have hidden features embedded in the noise, which can be identified by the GAN. That is why it is able to identify the sample classes without labels and generate their high-resolution spectra. Aspirin (black), ibuprofen (red), PCM (blue), ranitidine (green), 4-MBA (magenta), and 4-NTP (yellow).

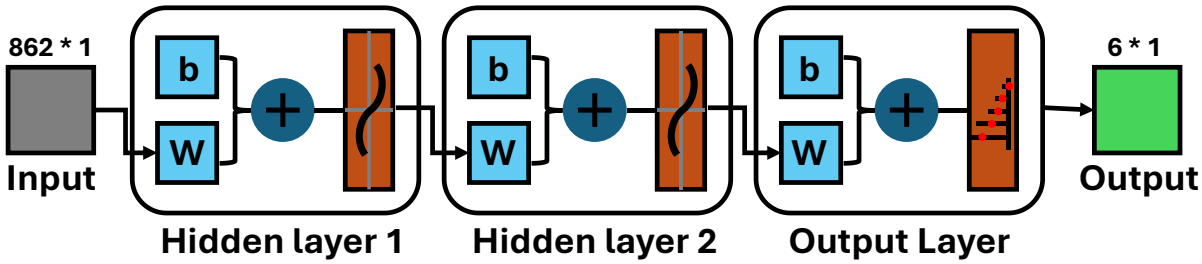

Figure S5: The architecture of the ANN classification network having highest accuracy and minimum test loss (model 1) is consists of one input layer, two hidden layers with 17 and 18 neurons respectively, followed by the classification layer in the end.

Table S3: The best three ANN classification network with their optimized parameters and performance during training and testing with additional GAN-generated high-resolution dataset.

Name	**Training Function**	**No. of hidden layers/No. of Neurons**	**Accuracy (Training, Testing, and Validation)**	**Accuracy (Additional dataset, GAN-generated)**	**Loss (Additional dataset, GAN-generated)**
Model 1	trainbr	[17, 18]	100 %	97.5 %	0.007922
Model 2	trainbr	[8, 16]	100 %	97.4 %	0.008250
Model 3	trainbr	[12, 20]	100 %	97.3 %	0.008335

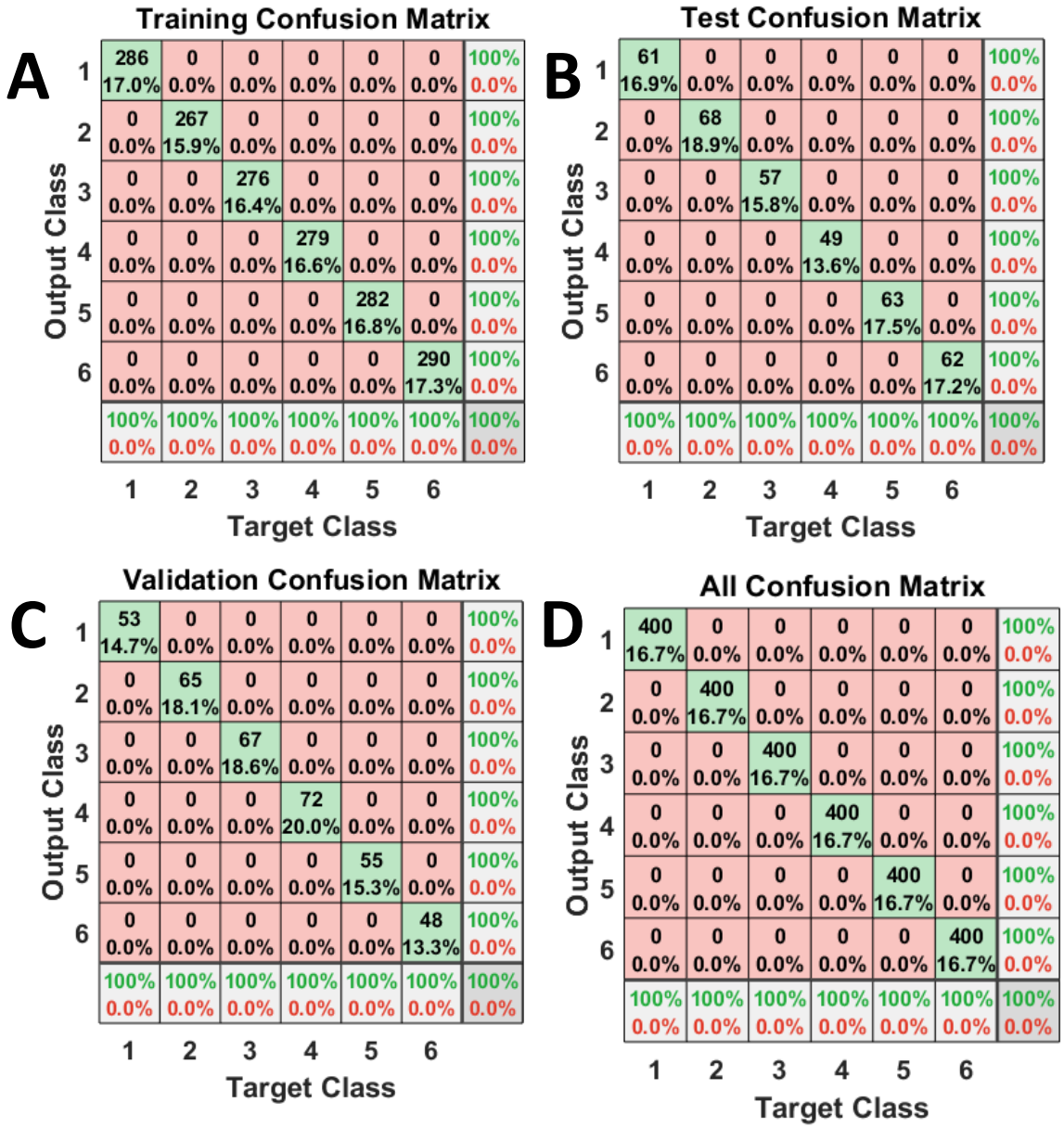

Figure S6: Confusion matrices showing the accuracy of the ANN classification network (model1) on high-resolution Raman dataset for (A) training, (B) testing, (C) validation, and (D) overall accuracy. The representation for different samples is as follows: (1) aspirin, (2) ibuprofen, (3) PCM, (4) ranitidine, (5) 4-MBA, and (6) 4-NTP. The confusion matrix shows the target class (actual) vs output class (predicted by the model). The similar kind of accuracy was observed for both model 2 and model 3.

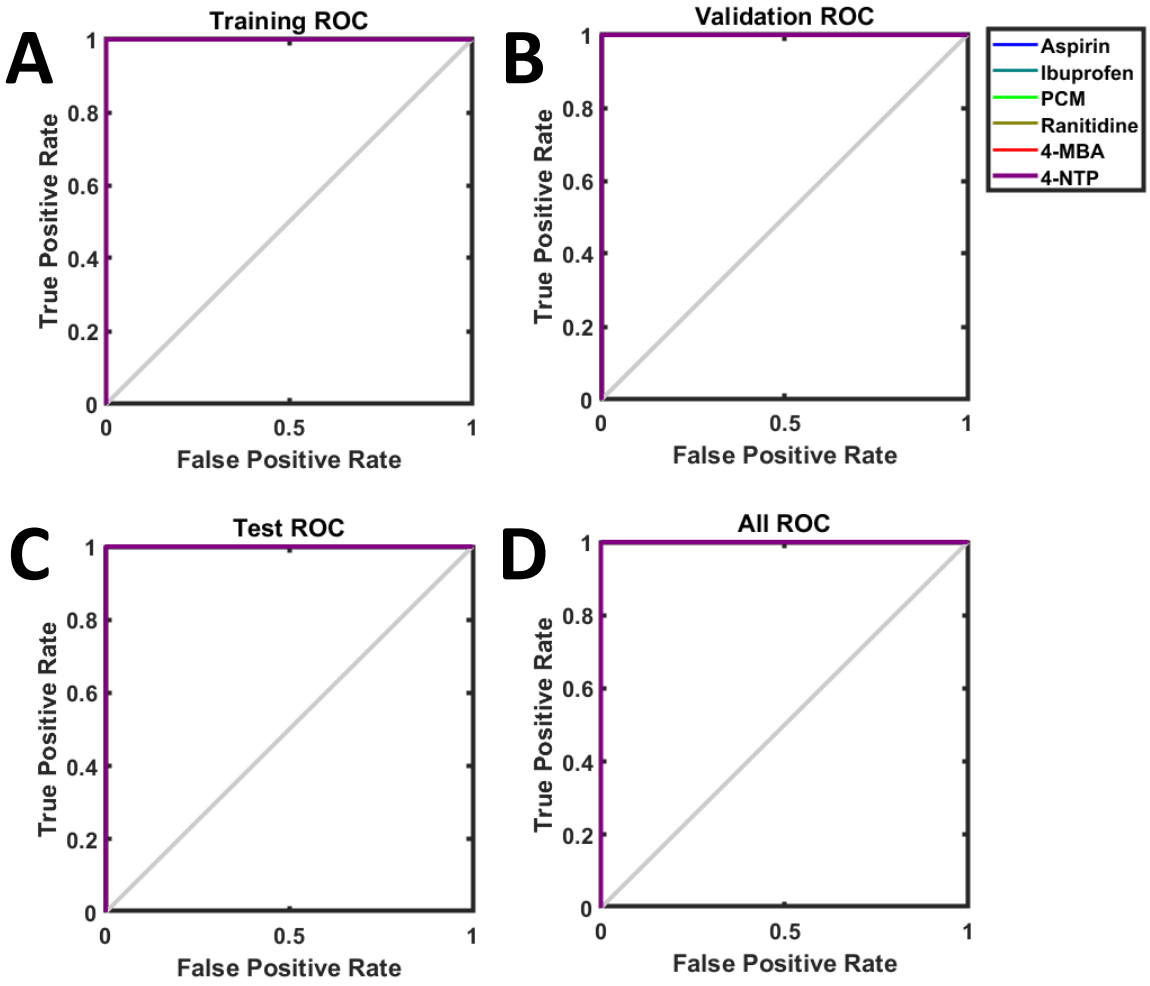

Figure S7: ROC of the ANN classification network (model1) on high-resolution Raman dataset for (A) training, (B) validation, (C) testing, and (D) overall. The closer the curve is to the one, the more accurate the results will be for classification. The similar kind of trend was observed for both model 2 and model 3.

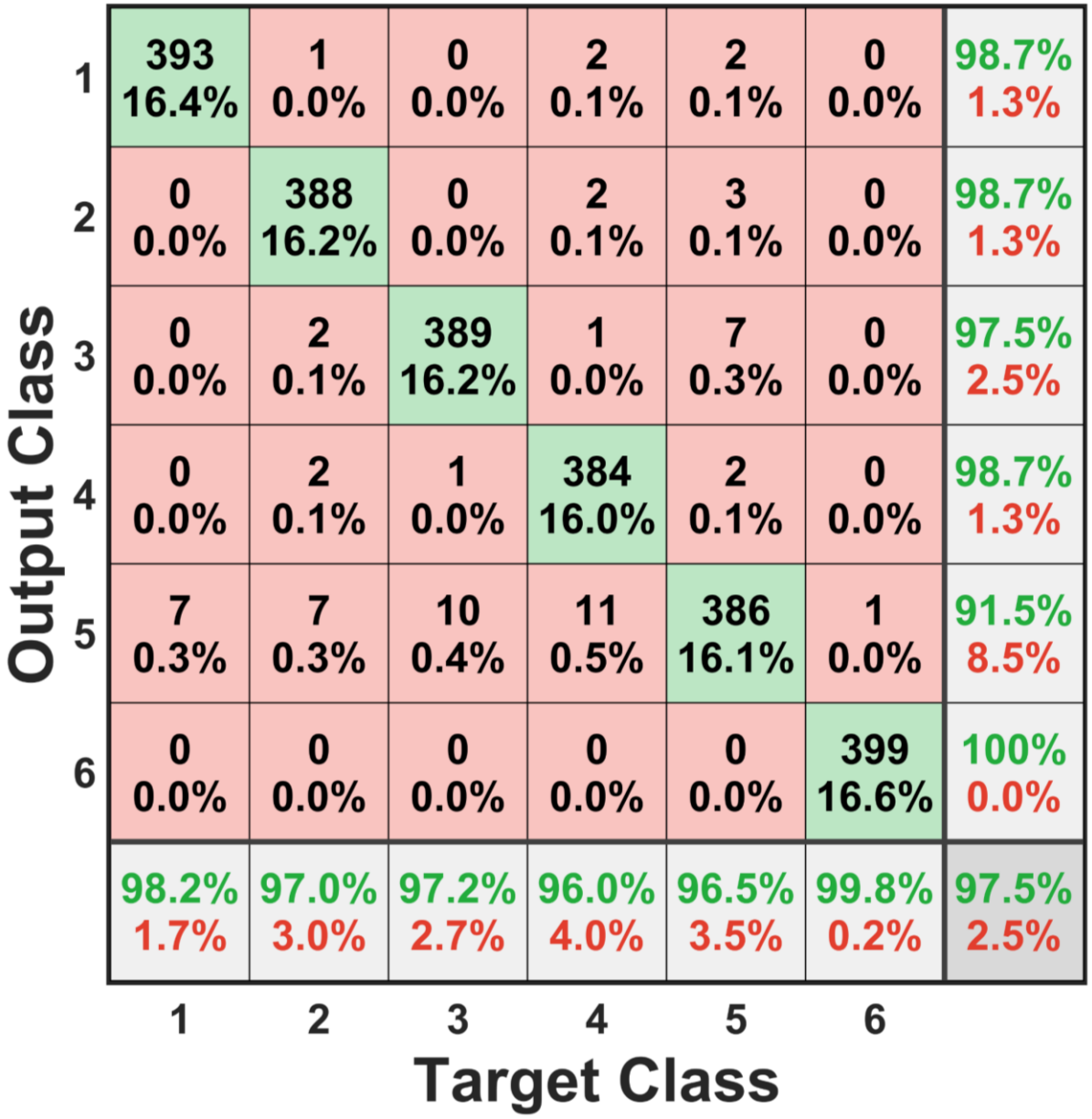

Figure S8: Confusion matrices showing the accuracy of the ANN classification network (model 1) for the additional dataset, i.e., GAN-generated high-resolution Raman data with a classification accuracy of 97.5%. The representation for different samples is as follows: (1) aspirin, (2) ibuprofen, (3) PCM, (4) ranitidine, (5) 4-MBA, and (6) 4-NTP.

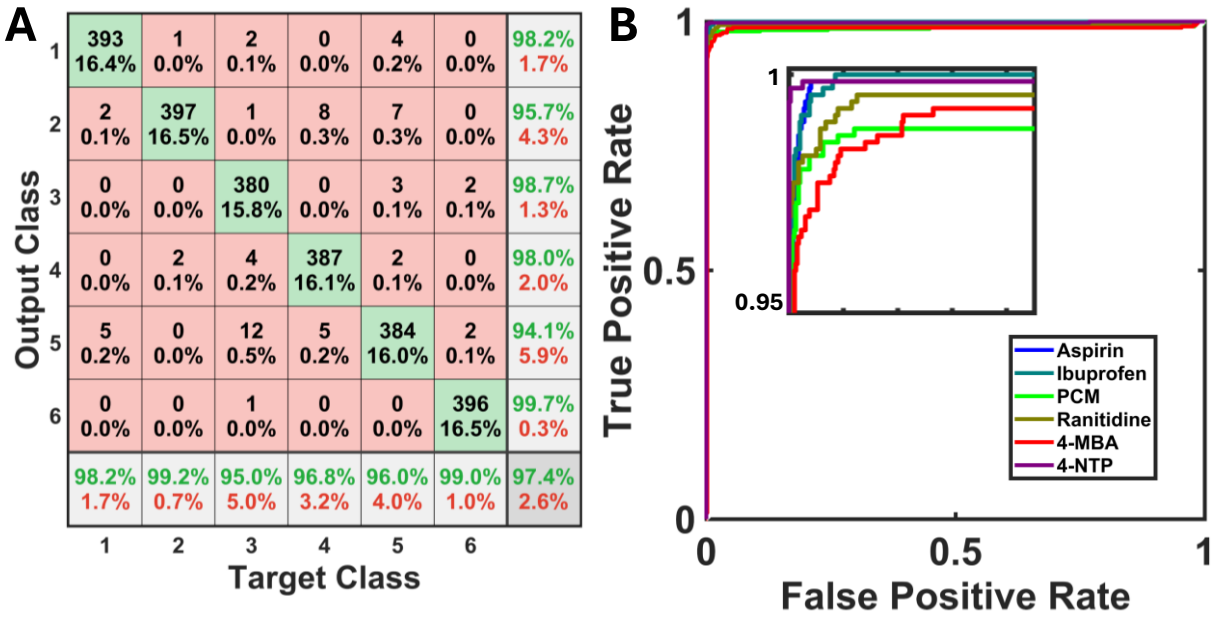

Figure S9: (A) Confusion matrices showing the accuracy of the ANN classification network (model 2) for the additional dataset, i.e., GAN-generated high-resolution Raman data with a classification accuracy of 97.4%. (B) ROC of the ANN classification network (model 2) for GAN-generated high-resolution Raman data. The closer the curve is to the one, the more accurate the results will be for classification. The representation for different samples is as follows: (1) aspirin, (2) ibuprofen, (3) PCM, (4) ranitidine, (5) 4-MBA, and (6) 4-NTP.

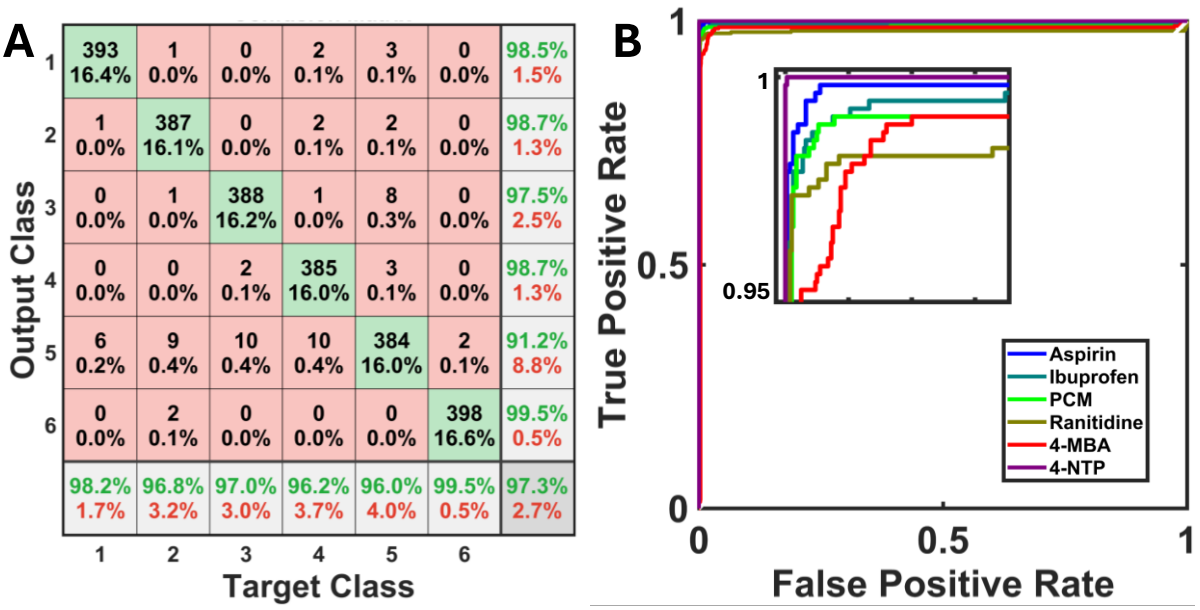

Figure S10: (A) Confusion matrices showing the accuracy of the ANN classification network (model 3) for the additional dataset, i.e., GAN-generated high-resolution Raman data with a classification accuracy of 97.3%. (B) ROC of the ANN classification network (model 3) for GAN-generated high-resolution Raman data. The closer the curve is to the one, the more accurate the results will be for classification. The representation for different samples is as follows: (1) aspirin, (2) ibuprofen, (3) PCM, (4) ranitidine, (5) 4-MBA, and (6) 4-NTP.

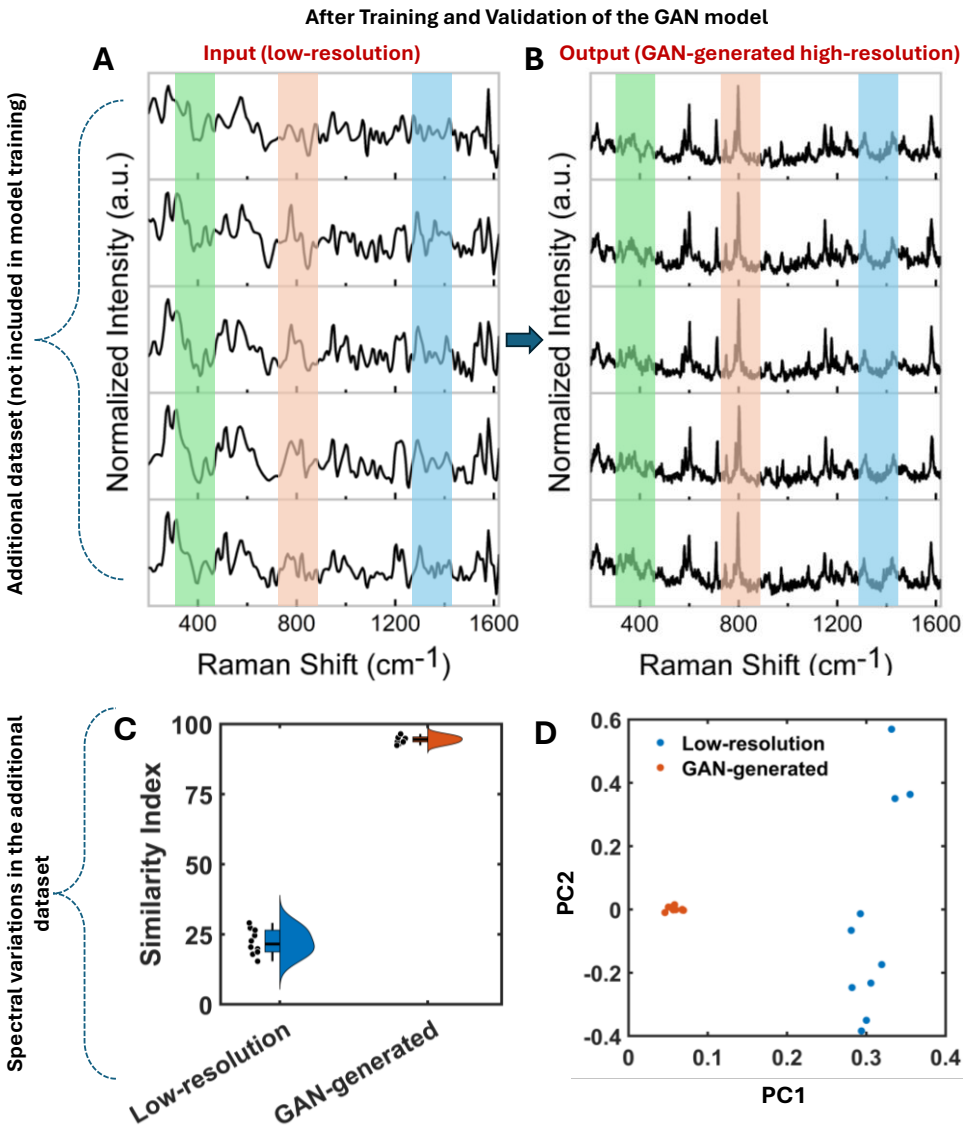

Figure S11: The GAN model was trained, validated, and tested using the 2400 low-resolution Raman spectra from 6 different class of samples (random splitting of dataset). The trained model for further tested on the additional low-resolution dataset (never fed to model) to check for the model accuracy to transform the low-resolution instance to a high-resolution Raman spectrum. The trained GAN model transformed the low-resolution to high resolution in completely unsupervised way. (A) The low-resolution Raman spectra of the ibuprofen drug, where the highlighted part in the low-resolution spectra shows the spectral differences (similarity index of 21% only) within the same sample under identical conditions. (B) This low-resolution Raman spectra (A) was fed to the trained GAN model to get the high-resolution instance. The highlighted part shows that instead of the spectra differences in low-resolution profile, the model capable to rebuild these lost features and generate the high-resolution counterpart (similarity index of 94%). (C) The violin plot shows the variation of similarity index in the additional low-resolution (10 more samples) with similarity index of around 21% and GAN-generated high resolution Raman spectra with similarity index of 94%. (D) The scatter plot from the PCA scores shows the highly random fluctuation (more spread in data points) in the low-resolution profile and minimum in the GAN-generated high-resolution Raman spectra.

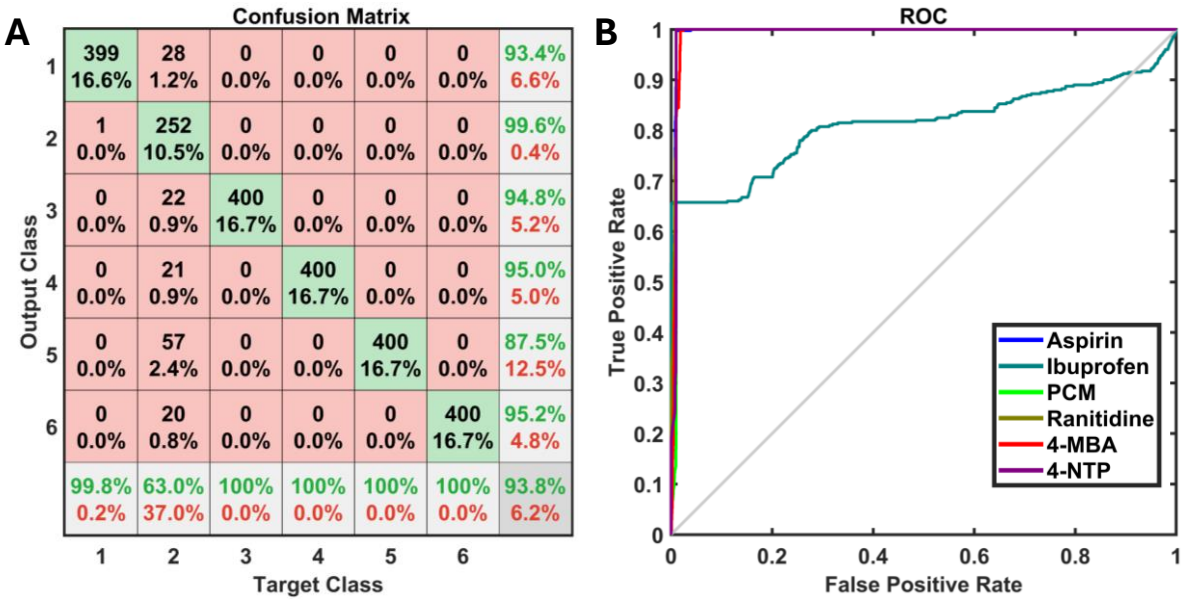

Figure S12: (A) Confusion matrices showing the accuracy of the ANN classification network for GAN-generated high-resolution Raman dataset in case of sample-out (ibuprofen sample is not a part of the training set during GAN model training) with a classification accuracy of 93.8%. (B) ROC of the ANN classification network for GAN-generated high-resolution Raman data. The closer the curve is to the one, the more accurate the results will be for classification. The representation for different samples is as follows: (1) aspirin, (2) ibuprofen, (3) PCM, (4) ranitidine, (5) 4-MBA, and (6) 4-NTP.

Table S4: SNR of Raman spectra accumulated using high-resolution (HR), low-resolution (LR), and machine learning (ML) generated for different samples.

Name	No. of Samples	Mean	Standard Deviation	Median	Minimum	Maximum
Aspirin HR	400	23.581	8.795	23.090	12.610	180.808
Aspirin LR	400	7.995	3.292	7.616	3.946	57.172
Aspirin ML	400	45.641	9.079	44.747	33.647	162.210
Ibuprofen HR	400	33.213	10.274	31.840	10.139	177.383
Ibuprofen LR	400	10.159	6.660	8.202	3.800	59.728
Ibuprofen ML	400	37.403	4.805	37.172	10.501	55.476
PCM HR	400	106.312	11.583	106.825	20.517	129.524
PCM LR	400	9.469	2.241	9.140	4.899	23.589
PCM ML	400	172.409	14.652	184.890	27.765	314.103
Ranitidine HR	400	61.021	11.661	60.635	22.270	104.754
Ranitidine LR	400	15.964	9.972	13.597	5.406	115.825
Ranitidine ML	400	118.224	17.320	117.960	41.314	187.577
4-MBA HR	400	83.942	12.214	84.013	38.648	126.680
4-MBA LR	400	14.466	5.201	13.521	6.171	39.749
4-MBA ML	400	133.958	15.229	143.864	21.522	232.148
4-NTP HR	400	85.252	18.124	34101.064	33.839	81.849
4-NTP LR	400	8.580	5.704	5.463	3.572	37.482
4-NTP ML	400	128.547	14.136	51419.117	4.889	13.985

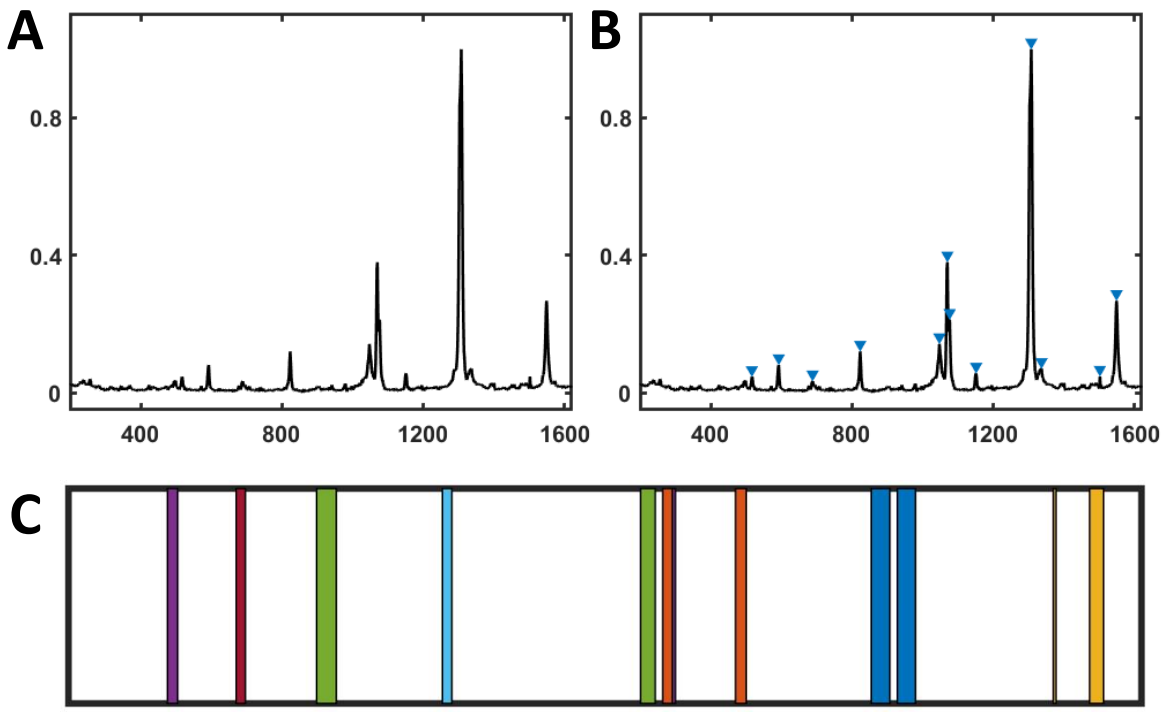

Figure S13: The process of making a Raman barcode. (A) the Raman spectra of the 4-NTP molecule. (B) using the findpeaks function to extract the peak location and FWHM and incorporate that information in the barcode. (C) The Raman barcode of the 4-NTP molecule.

Table S5: Average similarity index (%) of the low-resolution Raman barcode with the high-resolution Raman barcode.

Low / High	Aspirin	Ibuprofen	PCM	Ranitidine	4-MBA	4-NTP
Aspirin	**17.253**	20.468	18.523	14.258	16.425	18.097
Ibuprofen	16.811	**20.045**	18.144	12.733	14.987	16.933
PCM	18.052	20.481	**17.278**	14.023	15.810	17.511
Ranitidine	14.735	16.694	18.386	**15.277**	14.274	15.309
4-MBA	16.187	19.520	18.003	13.757	**13.717**	14.850
4-NTP	18.629	20.657	19.738	15.556	16.142	**18.462**

Table S6: Standard deviation of similarity index of the low-resolution Raman barcode with the high-resolution Raman barcode.

Low / High	Aspirin	Ibuprofen	PCM	Ranitidine	4-MBA	4-NTP
Aspirin	**7.027**	7.080	7.954	7.204	7.550	7.497
Ibuprofen	6.967	**7.740**	9.024	6.236	7.181	7.046
PCM	8.804	8.537	**7.889**	6.852	7.572	7.727
Ranitidine	6.474	7.666	8.095	**6.977**	7.076	7.139
4-MBA	8.592	9.021	9.035	7.214	**6.820**	7.744
4-NTP	8.131	8.491	8.798	7.615	7.923	**8.163**

Table S7: Average similarity index (%) of the GAN-generated high-resolution Raman barcode with the high-resolution Raman barcode.

GAN / High	Aspirin	Ibuprofen	PCM	Ranitidine	4-MBA	4-NTP
Aspirin	**97.643**	43.901	24.709	24.704	25.304	18.674
Ibuprofen	42.500	**95.250**	31.345	46.600	45.705	25.877
PCM	24.247	32.369	**97.344**	63.380	21.222	10.673
Ranitidine	33.932	48.212	64.693	**96.442**	44.400	60.332
4-MBA	26.253	44.157	22.200	40.674	**98.803**	50.456
4-NTP	18.518	26.869	11.376	12.235	45.305	**99.685**

Table S8: Standard deviation of similarity index of the GAN-generated high-resolution Raman barcode with the high-resolution Raman barcode.

GAN / High	Aspirin	Ibuprofen	PCM	Ranitidine	4-MBA	4-NTP
Aspirin	**8.555**	9.806	10.239	17.339	11.369	6.581
Ibuprofen	9.806	**11.832**	11.800	22.866	15.788	11.041
PCM	10.239	11.800	**15.786**	22.521	15.903	13.891
Ranitidine	17.339	22.866	22.521	**23.426**	19.815	23.693
4-MBA	11.369	15.788	15.903	19.815	**16.610**	16.229
4-NTP	6.581	11.041	13.891	23.693	16.229	**12.011**

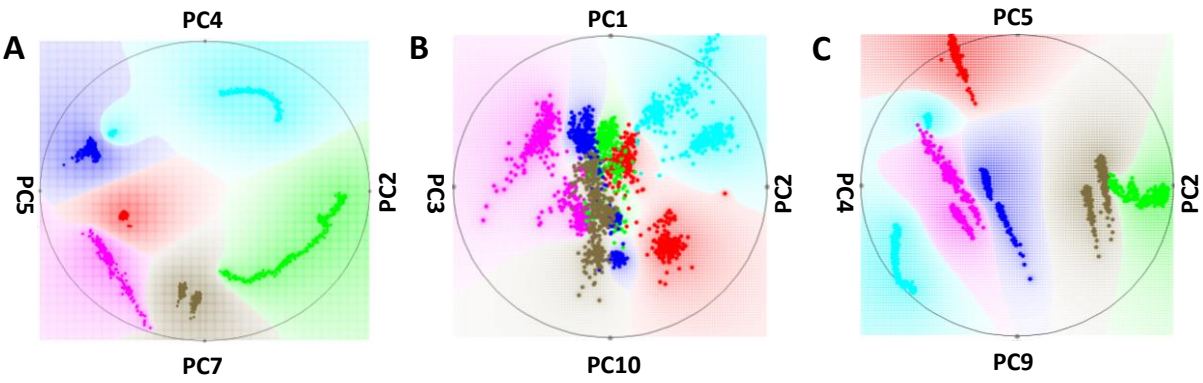

Figure S14: The RadViz clustering between different class of samples (A) high, (B) low-resolution, and (C) GAN-generated high-resolution Raman spectra showing different clustering efficiency. The clustering efficiency of GAN-generated high-resolution Raman spectra is significantly improved over its low-resolution counterpart. Aspirin (red), ibuprofen (blue), PCM (green), ranitidine (grey), 4-MBA (cyan), and 4-NTP (magenta).

REFERENCES

(1) Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A. A. Image-to-image translation with conditional adversarial networks. 2017, pp 1125-1134.
(2) Ma, X.; Wang, K.; Chou, K.; Li, Q.; Lu, X. Conditional Generative Adversarial Network for Spectral Recovery to Accelerate Single-Cell Raman Spectroscopic Analysis. *Analytical Chemistry* **2022**, *94* (2), 577-582, Article. DOI: 10.1021/acs.analchem.1c04263.
(3) Da, K. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.
(4) MACKAY, D. BAYESIAN INTERPOLATION. *Neural Computation* **1992**, *4* (3), 415-447, Note. DOI: 10.1162/neco.1992.4.3.415.
(5) Nasr, G. E.; Badr, E. A.; Joun, C. Cross entropy error function in neural networks: Forecasting gasoline demand. 2002, pp 381-384.
(6) Aurelio, Y. S.; De Almeida, G. M.; de Castro, C. L.; Braga, A. P. Learning from imbalanced data sets with weighted cross-entropy function. *Neural processing letters* **2019**, *50*, 1937-1949.
(7) Qi, Y.; Hu, D.; Jiang, Y.; Wu, Z.; Zheng, M.; Chen, E.; Liang, Y.; Sadi, M.; Zhang, K.; Chen, Y. Recent Progresses in Machine Learning Assisted Raman Spectroscopy. *Advanced Optical Materials* **2023**, *11* (14), Review. DOI: 10.1002/adom.202203104.
(8) Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. Image quality assessment: From error visibility to structural similarity. *Ieee Transactions on Image Processing* **2004**, *13* (4), 600-612, Article. DOI: 10.1109/TIP.2003.819861.
(9) Pan, L.; Pipitsunthonsan, P.; Zhang, P.; Daengngam, C.; Booranawong, A.; Chongcheawchamnan, M. Noise reduction technique for Raman spectrum using deep learning network. 2020, IEEE: pp 159-163.
(10) Kassu, A.; Robinson, D. Vibrational Spectroscopy of Pain Relievers: Traditional and Remote Raman Techniques. *Journal of Analytical Sciences, Methods and Instrumentation* **2023**, *13* (3), 27-37.
(11) Ramesh, P.; Gunasekaran, S.; Ramkumar, G. R. Molecular Structure, Vibrational Spectra, UV-Visible and NMR Spectral Analysis on Ranitidine Hydrochloride using AB Initio and DFT Methods. *International Journal of Current Research and Academic Review* **2015**, *3* (11), 117-138.
(12) Thorley, F.; Baldwin, K.; Lee, D.; Batchelder, D. Dependence of the Raman spectra of drug substances upon laser excitation wavelength. *Journal of Raman Spectroscopy* **2006**, *37* (1-3), 335-341, Article. DOI: 10.1002/jrs.1446.
(13) Chen, S.; Fan, J.; Lv, M.; Hua, C.; Liang, G.; Zhang, S. Internal Standard Assisted Surface-Enhanced Raman Scattering Nanoprobe with 4-NTP as Recognition Unit for Ratiometric Imaging Hydrogen Sulfide in Living Cells. *Analytical Chemistry* **2022**, Article|Early Access. DOI: 10.1021/acs.analchem.2c02961.
(14) Marques, F.; Alves, R.; dos Santos, D.; Andrade, G. Surface-enhanced Raman spectroscopy of one and a few molecules of acid 4-mercaptobenzoic in AgNP enabled by hot spots generated by hydrogen bonding. *Physical Chemistry Chemical Physics* **2022**, *24* (44), 27449-27458, Article. DOI: 10.1039/d2cp03375e.