Evaluation of Spectral Retrieval Methods for Hyperspectral Coherent Anti-Stokes Raman Scattering Microscopy

A Thesis Submitted to the Committee of Graduate Studies in
Partial Fulfillment of the Requirements of the Degree of

Master of Science

in the Faculty of Arts and Science

TRENT UNIVERSITY

Peterborough, Ontario, Canada

Materials Science M.Sc. Graduate Program

September 2023

Abstract

Evaluation of Spectral Retrieval Methods for Hyperspectral Coherent Anti-Stokes Raman Scattering Microscopy

John Shafe-Purcell

Coherent anti-Stokes Raman scattering (CARS) microscopy is a label-free chemical imaging modality that uses CARS as a contrast mechanism to spatially resolve materials based on their molecular vibrational spectra. Due to the presence of a non-resonant background that obfuscates the chemical information contained in CARS spectra, CARS images suffer from poor contrast and cannot be readily used for quantitative chemical analysis. Over the past two decades, spectral retrieval methods have been developed to obtain Raman-like spectra from CARS spectra. These methods promise to improve image contrast and enable reliable quantitative analysis. In this work I systematically evaluate a selection of the forefront spectral retrieval methods, including both analytical and machine learning approaches, to determine how well they perform at the task of non-resonant background removal. The more recent machine learning methods demonstrate remarkable performance on spectra resembling the training dataset but are not as suitable as the analytical methods in general. The analytical methods based on the discrete Hilbert transform thus remain preferable due to their ease-of-use and general applicability.

Keywords: coherent anti-Stokes Raman scattering, spectral phase retrieval, non-resonant background, chemical imaging, hyperspectral imaging, Kramers-Kronig analysis, machine learning.

Acknowledgments

I would like to thank…

- Dr. Aaron Slepkov for being my research supervisor and introducing me into the world of nonlinear optics and chemical imaging.
- Joel Tabarangao, Jeremy Porquez, and Ryan Cole for their work developing the CARS microscopy system at Trent University. Without their previous work, getting into CARS microscopy would have been far more difficult. When I was first getting started, Jeremy provided me with all the guidance I needed that was essential for my quick integration into the lab.
- George Olaniyan for introducing me to the deep learning techniques for non-resonant background removal, namely SpecNet and VECTOR. Learning about these methods, and the lack of any clear guidance on which are best or how to use them, inspired my desire to pursue a project comparing the currently available NRB removal methods. This has since grown into the topic of this thesis.
- Trent University for granting me the Graduate Research Fellowship Award that helped fund my graduate research.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CAE *Convolutional Autoencoder*

A type of neural network architecture called an "autoencoder" which utilizes convolutional layers in the encoder and transposed convolutional layers in the decoder.

CARS *Coherent Anti-Stokes Raman Scattering*

A coherent Raman scattering process that occurs when pump/probe and Stokes photons interact with a Raman-active material, producing higher-frequency anti-Stokes photons when the energy difference between the pump/probe and Stokes photons matches a vibrational resonance.

CNN *Convolutional Neural Network*

A neural network containing convolutional layers. These are layers that learn convolution filters/kernels during training which are convolved with the input data during forward propagation.

DFG *Difference-frequency Generation*

Second-order nonlinear optical effect whereby two photons interact and produce a third photon with an energy (frequency) that is the difference between the two interacting photons.

DHT *Discrete Hilbert Transform*

An algorithm for computing the Hilbert transform of a discrete function, where the Hilbert transform and its inverse are functionally equivalent to the Kramers-Kronig relations.

DL *Deep Learning*

A machine learning approach that utilizes artificial neural networks and mathematical optimization techniques for a machine to "learn" to perform a task. The "deep" in deep learning refers to the considerable number of layers in the neural network.

FWHM *Full Width at Half-maximum*

The full width of a peak at half of its amplitude.

FWM *Four-wave mixing*

Third-order nonlinear optical effect involving four photons.

KK *Kramers-Kronig*

Relating to the Kramers-Kronig spectral phase retrieval method for non-resonant background removal. This method uses the Kramers-Kronig relations to obtain an equation for the phase of the complex-valued third-order susceptibility in order to retrieve a Raman-like spectra from CARS.

LeDHT *"Learned Discrete Hilbert Transform"*

An approach for calculating the discrete Hilbert transform using a transformation matrix obtained from machine learning techniques.

ML *Machine Learning*

Any process by which a machine gradually improves its performance at a given task without being given an explicit algorithm for accomplishing that task.

NRB *Non-resonant Background*

Vibrationally non-resonant four-wave mixing that produces an effective background that distorts the CARS signal.

SFG *Sum-frequency Generation*

Second-order nonlinear optical effect whereby two photons interact and produce a third photon with an energy (frequency) that is the sum of the two interacting photons.

SHG *Second Harmonic Generation*

Special case of SFG where the two photons have equal energy.

SpecNet *"Spectral Retrieval Convolutional Neural Network"*

A convolutional neural network architecture designed for CARS non-resonant background removal.

THG *Third Harmonic Generation*

Special case of FWM where three photons of equal energy interact to produce a fourth photon with three times the energy of the incident photons.

VECTOR *"Very Deep Convolutional Autoencoder"*

A convolutional autoencoder architecture designed for CARS non-resonant background removal.

$\mathbf{cm^{-1}}$ *Wavenumber*

Metric unit representing "spatial frequency", commonly used in spectroscopy to represent energy transitions. Given by:

$$\frac{1}{\lambda} \times 10^{-7}$$

Where λ (in units of nanometers) is the wavelength corresponding to a photon with the given energy.

1 Introduction

This section will introduce the reader to coherent anti-Stokes Raman scattering (CARS), why it is a desirable contrast mechanism for label-free hyperspectral microscopy, and the preeminent issue facing the widespread adoption of CARS microscopy, namely, the non-resonant background (NRB). This will include a brief introduction to nonlinear optics, how CARS and the NRB originate from nonlinear optics, and the NRB removal problem. The overall goal of this thesis is to investigate the available NRB removal methods and demonstrate through a comparative analysis the capabilities and shortcomings of each. This thesis should provide the reader with the knowledge to make an informed decision about which currently available NRB removal method to employ for their CARS application.

1.1 Nonlinear Optics

When an electric field interacts with dielectric materials, the induced polarization P of the material can be given by the Taylor expansion in terms of the incident electric field E as:

$$P = \epsilon_0\left(\chi^{(1)}E + \chi^{(2)}E^2 + \chi^{(3)}E^3 + \cdots\right) \quad (1)$$

Here, the coefficients given by $\chi^{(n)}$ are the n-th order susceptibilities of the material and $\epsilon_0 = 8.854\ \mathrm{F/m}$ is the vacuum permittivity [1]. It is important to note that this induced polarization refers to the dipole moments created by the separation of charge in the material as a response to the incident electric field, not the polarization of the incident electric field itself. Since light is composed of the synchronized oscillation of electric and magnetic fields as described by Maxwell's equations, when light interacts with a material it will induce an oscillatory polarization within the material as a result. This time-varying polarization will produce a new electromagnetic wave because of this interaction. This is

referred to as the scattered or produced light. When the electric field intensity is small, like that generated by continuous-wave lasers and incoherent light sources, the first order $\chi^{(1)}$ term dominates. This produces linear optical effects where the induced polarization is proportional to the incident electric field. Advances in ultrafast laser technologies has allowed us to create pulses of light with extremely high peak powers. These lasers can generate instantaneous electric field intensities large enough that higher order $\chi^{(2)}$ and $\chi^{(3)}$ effects become significant, giving rise to the field of nonlinear optics [2].

Second-order nonlinear effects are those for which the $\chi^{(2)}$ term dominates, and so the polarization becomes $P = \epsilon_0 \chi^{(2)} E_1 E_2$. Here, E^2 from Eq. (1) is replaced with two separate electric fields, E_1 and E_2, because two independent electric fields can interact with the material simultaneously. Second-order effects notably include *sum*- and *difference-frequency generation* (SFG and DFG), for which a third photon is produced with an energy that is the sum or difference between the energies of two incident photons, respectively. *Second harmonic generation* (SHG) is a special case of SFG where two photons with equal energies produce a third photon with double the energy of the incident photons.

Third-order effects are those for which the $\chi^{(3)}$ term dominates, and so the polarization becomes $P = \epsilon_0 \chi^{(3)} E_1 E_2 E_3$. Here, E^3 from Eq. (1) is replaced with three separate electric fields, E_1, E_2, and E_3, because three independent electric fields can interact with the material simultaneously. *Four-wave mixing* (FWM) is the most relevant third-order optical effect for this work. As the name suggests, it involves four electromagnetic waves, where two or three incident waves produce two or one new waves, respectively. *Third harmonic generation* (THG) is a special case of FWM where three incident photons with equal energy produce a fourth photon with triple the energy of the three incident photons.

Coherent anti-Stokes Raman scattering (CARS) is a special case of FWM where three photons of various energies produce a fourth photon with a higher energy when the energy difference between any two of the incident photons corresponds to a molecular vibrational resonance. Nonlinear optical processes are phase-sensitive and require proper phase matching conditions to be met for efficient generation. CARS also requires that Raman selection criteria be satisfied. Figure 1 below shows the schematic Jablonski diagrams for the nonlinear optical processes discussed in this section.

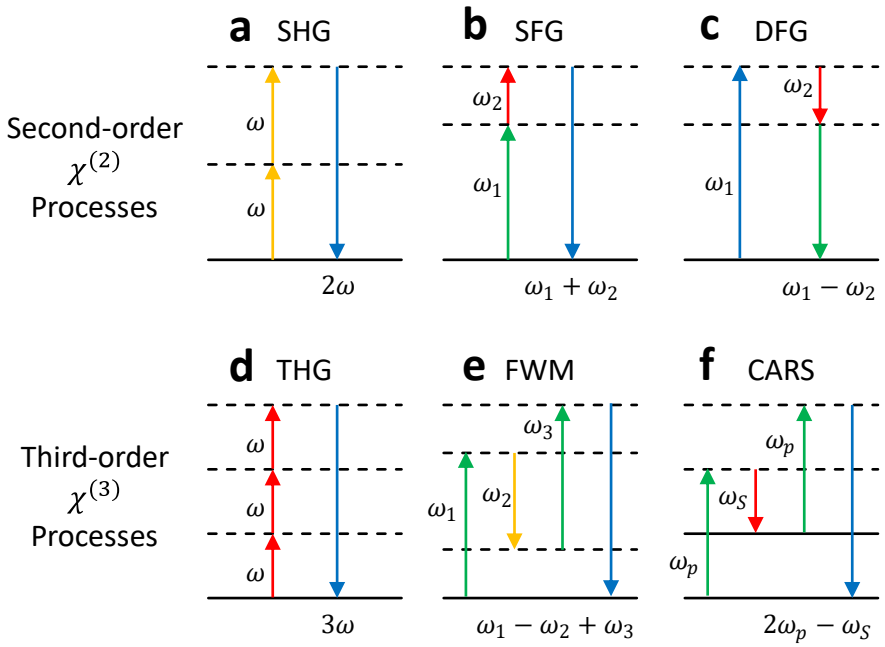

Figure 1: Simplified Jablonski diagrams for various nonlinear optical effects. The diagrams are given for: (a) Second-harmonic generation (SHG), (b) sum-frequency generation (SFG), (c) difference-frequency generation (DFG), (d) third-harmonic generation (THG), (e) non-resonant four-wave mixing (FWM), and (f) degenerate coherent anti-Stokes Raman scattering (CARS). Virtual energy levels are represented by black dashed lines, whereas the ground state and vibrational energy levels are represented by black solid lines.

1.2 Coherent Anti-Stokes Raman Scattering

CARS, as a special case of FWM, is a third-order nonlinear optical effect that involves the interaction of three incident laser beams—the pump, probe, and Stokes beams—and produces a fourth beam of higher energy, called the anti-Stokes beam. When the incident beams interact with the scatterers in the excitation volume, they induce the formation of an ensemble of oscillating Hertzian dipoles. These can be modelled classically as damped harmonic oscillators driven by the coherent addition of the probe beam and the difference frequency between the pump and Stokes beams. The anti-Stokes beam is then given by the far-field radiation produced by these oscillators [3]–[6]. Since CARS is a $\chi^{(3)}$ effect, the electric field corresponding to the anti-Stokes emission can be written as:

$$E_{CARS}(\omega_\mathrm{p} - \omega_\mathrm{S} + \omega_\mathrm{pr}) = \chi^{(3)} E_\mathrm{p}(\omega_\mathrm{p}) E_\mathrm{pr}(\omega_\mathrm{pr}) E_\mathrm{S}(\omega_\mathrm{S}) \tag{2}$$

where E_p, E_pr, and E_S are the electric fields and ω_p, ω_pr, and ω_S are the frequencies of the pump, probe, and Stokes beams, respectively. The emitted anti-Stokes beam has an electric field E_{CARS} with a frequency given by $\omega_{AS} = \omega_\mathrm{p} - \omega_\mathrm{S} + \omega_\mathrm{pr}$. In most practical cases, the pump and probe beams come from the same source and thus have the same energy (i.e. $\omega_\mathrm{pr} = \omega_\mathrm{p}$). This is called degenerate CARS. The previous equation can be rewritten for degenerate CARS as:

$$E_{CARS}(2\omega_\mathrm{p} - \omega_\mathrm{S}) = \chi^{(3)} E_p^2(\omega_\mathrm{p}) E_S(\omega_\mathrm{S}) \tag{3}$$

Where the frequency of the anti-Stokes photon is now $\omega_\mathrm{AS} = 2\omega_\mathrm{p} - \omega_\mathrm{S}$. The term CARS will refer to this degenerate case for the remainder of this work. The CARS **intensity** is then given by,

$$I_{CARS} \propto \left|\chi^{(3)}\right|^2 I_p^2 I_S \tag{4}$$

This shows that the CARS intensity is proportional to the modulus squared of the total third-order susceptibility, the squared pump beam intensity, and the Stokes beam intensity.

The third-order susceptibility, $\chi^{(3)}$, can be modelled as the sum of two components; one that arises from the vibrational resonances of the material, $\chi_R^{(3)}$, and one that arises from the non-resonant FWM, $\chi_{NR}^{(3)}$, as follows:

$$\chi^{(3)} = \chi_R^{(3)} + \chi_{NR}^{(3)} \tag{5}$$

Assuming that the pump and Stokes intensities are held constant, the two previous equations can be combined to give the following proportionality:

$$I_{CARS} \propto \left|\chi_R^{(3)} + \chi_{NR}^{(3)}\right|^2 \tag{6}$$

Given that the non-resonant contribution $\chi_{NR}^{(3)}$ is strictly real-valued, which is an empirically valid assumption in most cases [7], the above can be expanded to give:

$$I_{CARS} \propto \left|\chi_{\mathrm{R}}^{(3)}\right|^2 + \left(\chi_{\mathrm{NR}}^{(3)}\right)^2 + 2\mathrm{Re}\left(\chi_{\mathrm{R}}^{(3)}\right)\chi_{\mathrm{NR}}^{(3)} \tag{7}$$

The *non-resonant background* (NRB) is caused by the two terms containing $\chi_{\mathrm{NR}}^{(3)}$. This non-resonant contribution does not inform us about the molecular vibrational resonances but does act to obfuscate the chemical information encoded by the resonant contribution. Because of the cross-term, there is clearly no trivial way to decouple the resonant and non-resonant contributions through subtraction or factorization, which makes it difficult to use CARS for chemical analyses.

1.3 CARS Hypermicroscopy

Referring to Figure 1, panels (e) and (f) show the Jablonski diagrams for FWM and CARS, respectively. The frequency difference between the pump and Stokes, referred to as the Raman shift, is given by:

$$\omega = \omega_{\mathrm{p}} - \omega_{S} \tag{8}$$

You may recognise this as the beat frequency between the pump and Stokes beams that drives the molecular vibration. By varying the difference between the frequencies of the pump and Stokes beams, we can scan ω and collect a spectrum of the anti-Stokes emission at $\omega_{\mathrm{AS}} = 2\omega_{\mathrm{p}} - \omega_{\mathrm{S}}$ as a function of ω. This allows us to collect a so-called CARS spectrum. When ω corresponds to a molecular vibrational resonance, Ω, the oscillators are driven more efficiently, and the resonant part of the anti-Stokes signal is enhanced. However, as we scan through ω and record the anti-Stokes emission, we are simultaneously recording a spectrum of the non-resonant FWM as well. These resonant and non-resonant contributions to the susceptibility are the inspiration for Eq. (5).

Hyperspectral images contain both spatial and spectral data. Each hyperspectral image can be thought of as a three-dimensional data cube constructed by taking a spectrum at each pixel of an image. One dimension corresponds to the spectral axis and the other two dimensions correspond to the spatial axes. When we apply hyperspectral imaging to microscopy, we may call it hyperspectral microscopy or “hypermicroscopy”. CARS hypermicroscopy allows us to resolve spatial and spectral features simultaneously by using the CARS signal intensity as a contrast mechanism while imaging. A given frame in the hyperspectral image “stack” will correspond to a particular molecular vibrational resonance where brighter regions will correspond to a higher CARS intensity and darker

regions will correspond to a lower CARS intensity. This creates a sort of chemical contrast allowing us to spatially map chemical species to specific image regions. However, the NRB complicates the interpretation of these data.

Due to the presence of the NRB, the CARS signal is not strictly quadradic in analyte concentration. Consequently, quantitative analysis cannot be performed on CARS spectra as readily as it can with spontaneous Raman. Moreover, the dispersive effects corresponding to the real part of the resonant susceptibility cause the tail ends of the resonance peaks to extend far across the spectra. In hyperspectral imaging, this creates the appearance of chemical contrast where there are no Raman peaks corresponding to molecular vibrational resonances, as demonstrated in Figure 2 below.

Despite the NRB being a barrier to the adoption of CARS as a reliable method for chemical analysis, CARS offers undeniable benefits over competing techniques. For example, CARS offers a signal amplification up to 100 times greater than spontaneous Raman due to the coherence of the CARS process, allowing for shorter acquisition times and faster imaging speeds [8]. Also, the simplicity of CARS relative to other coherent Raman techniques means that it is cheaper and easier to deploy in most cases. Consequently, the development of techniques to effectively remove the NRB are necessary to unlock the full potential of CARS and remove barriers to its widespread adoption.

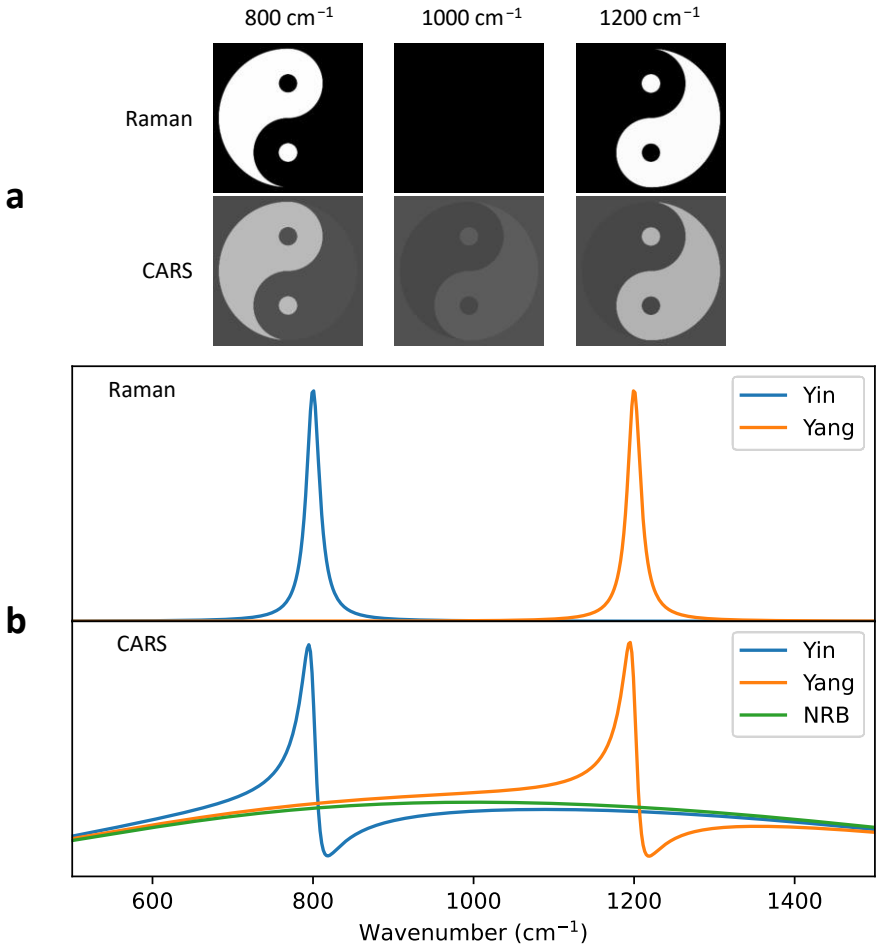

Figure 2: Effect of the NRB on hyperspectral images. This figure demonstrates how the NRB obfuscates the chemical information and creates chemical contrast where there are no corresponding molecular vibrational resonances. To demonstrate this, two distinct image regions have been chosen, one represented by the Yin and the other the Yang. The Yin represents a chemical with a single resonance peak at 800 cm^{-1} and the Yang presents a chemical with a single resonant peak at 1200 cm^{-1}, each with a half-width of 10 cm^{-1}. (a) shows the frames taken from the hyperspectral Raman (top) and CARS (bottom) images at 800 cm^{-1} (left), 1000 cm^{-1} (middle), and 1200 cm^{-1} (right). (b) shows the Raman (top) and CARS (bottom) spectra taken from the Yin and Yang regions. From (a) we can visually see the spurious chemical contrast in the CARS images, while (b) shows how this arises from the NRB. The horizontal axis of each plot in (b) represents zero intensity. This figure was created using the hyperspectral image simulation method described in Section 2.2.4.

1.4 Non-Resonant Background Removal

NRB "removal" is not simply a process of subtracting an additive background like one would do with the fluorescence background in spontaneous Raman spectroscopy. Instead, NRB removal is a process that involves retrieving the complex phase of the third-order susceptibility so that we may obtain the Raman spectral line shapes encoded in the imaginary part. Complex values can be represented by an expression of the form,

$$Ae^{i\phi} \tag{9}$$

Where A is the amplitude and ϕ is the phase of the value in the complex plane. From this, we can express the complex valued third-order susceptibility as,

$$\chi^{(3)}(\omega) = \left|\chi^{(3)}(\omega)\right| e^{i\phi(\omega)} \tag{10}$$

Where $\left|\chi^{(3)}(\omega)\right|$ is the amplitude and $\phi(\omega)$ is the phase. As mentioned previously, the measurable CARS signal intensity is given by $I_{CARS} \propto \left|\chi^{(3)}\right|^2$. This means that the amplitude of our susceptibility in the complex plane is proportional to the square root of the measured CARS signal, $A \propto \sqrt{I_{CARS}}$. This makes completely solving for $\chi^{(3)}(\omega)$ from measured CARS spectra simply a matter of finding the phase, $\phi(\omega)$. If the phase is known, the corresponding Raman spectrum is given by,

$$I_{Raman}(\omega) = \mathrm{Im}\left(\chi^{(3)}(\omega)\right) = \mathrm{Im}\left(\left|\chi^{(3)}(\omega)\right| e^{i\phi(\omega)}\right) \tag{11}$$

Spectral phase retrieval is thus the principal issue regarding NRB removal in CARS spectroscopy and hypermicroscopy. The rest of this work will evaluate several of these spectral retrieval methods to determine how well each method removes the NRB and whether the NRB problem has been essentially solved.

2 Methods

In this section I will outline the spectral retrieval methods assessed later in this work, the methodologies for data simulation, and evaluation metrics used to compare the retrieval methods.

2.1 Spectral Retrieval Methods

Several methods have been developed for retrieving the Raman-like spectral line shapes from CARS spectra, effectively "removing" the NRB. These promise to improve the analytical interpretability of CARS spectra by disentangling the underlying chemical information from the NRB. This section will provide an overview of four post-processing based NRB removal techniques. These were chosen to be included in this thesis because they represent the current state-of-the-art in NRB removal within the CARS literature. The overview for each method will include a brief description of the theory of operation, the general implementation details, and the potential advantages/disadvantage of the method. Each of these methods are freely available to use, modify, and publish under their respective open-access licenses, so the reader is encouraged to investigate these methods themselves for additional details.

2.1.1 Kramers-Kronig Spectral Phase Retrieval

The time-domain *Kramers-Kronig* (KK) method for spectral phase retrieval was developed in 2009 by Liu *et. al.* [9] and has been used extensively for NRB removal since its inception. Its creation was motivated by the fact that taking the natural logarithm of both sides of Eq. (10) gives,

$$\ln\left(\chi^{(3)}(\omega)\right) = \ln\left|\chi^{(3)}(\omega)\right| + i\phi(\omega) \tag{12}$$

This equation clearly satisfies the general form of a complex valued function for which we can apply the following Kramers-Kronig relation [10], [11],

$$\phi(\omega) = -\frac{P}{\pi}\int_{-\infty}^{\infty} \frac{\ln\left|\chi^{(3)}(\omega')\right|}{\omega' - \omega}\,\mathrm{d}\omega' \tag{13}$$

Where P is the Cauchy principal value. The article describing the KK method derived the relationship between the Kramers-Kronig relation and the Fourier transform, allowing it to be implemented using the discrete Fourier transform with imposed causality conditions [9]. An equivalent but simplified approach has since been adopted that uses the *discrete Hilbert transform* (DHT) as follows [12]:

$$\phi_{CARS/NRB}(\omega) = \mathcal{H}\left(\frac{1}{2}\ln\frac{I_{\mathrm{CARS}}(\omega)}{I_{\mathrm{NRB}}(\omega)}\right) \tag{14}$$

Where $\mathcal{H}$ is the DHT, I_{CARS} is the known CARS signal, and I_{NRB} is the known NRB signal. The NRB in this case is used as an internal reference by which we normalize the CARS signal to remove any system responses.

The KK method is an analytical technique that is simple and computationally efficient due to the utilization of the fast Fourier transform, but suffers from two major limitations: the underlying DHT returns significant errors for any **finite discrete** spectral domain because the Hilbert transform is only well-defined for an infinite continuous domain, and an accurate NRB profile (absent of resonant peaks) must be known. Improvements in the accuracy of this method have been achieved through error correction measures [12]. These can be summarized as follows:

1. Phase error correction via baseline detrending. The retrieved phase should have a zero baseline, but often has a slowly varying baseline that arises due to computational errors. The asymmetric least squares (ALS) baseline removal algorithm [13] can be used find the baseline and subtract it form the retrieved phase. This alone significantly enhances the accuracy of the KK method.
2. Scale error correction via unity centering of the real component of the retrieved phase-corrected spectrum. The real component of the previously retrieved phase-corrected spectrum should be unity centered, i.e. $\langle I_{\mathrm{CARS}}/I_{\mathrm{NRB}}\, cos\phi_{\mathrm{CARS/NRB}}\rangle = 1$. Any deviation of the mean trendline from 1 is due to scale errors that can be factored out so that the spectrum becomes unity centered. A Savitzky-Golay filter [14] can be used to find a mean trendline in the real component by which we divide the complex retrieved spectrum in an elementwise manner to correct the scale error.

Although these corrections significantly improve the accuracy of the KK method, it is still limited by the need for knowledge of the NRB spectrum. Typically, a surrogate NRB is experimentally obtained by taking a CARS spectrum of a material with few Raman bands, such as glass or water. However, even water and glass have Raman active bands that can cause errors in the retrieval depending on which transition energies are being probed [15], [16]. Another practical issue with the KK method is the potential for undefined behaviour in Eq. (14) due to division-by-zero or negative logarithm inputs for $I_{NRB} \leq 0$ or $I_{CARS} < 0$. So, care must be taken to clean the CARS and NRB spectra prior to using the KK method to analyze them. Figure 3 below shows a flow diagram summarizing the KK method procedure.

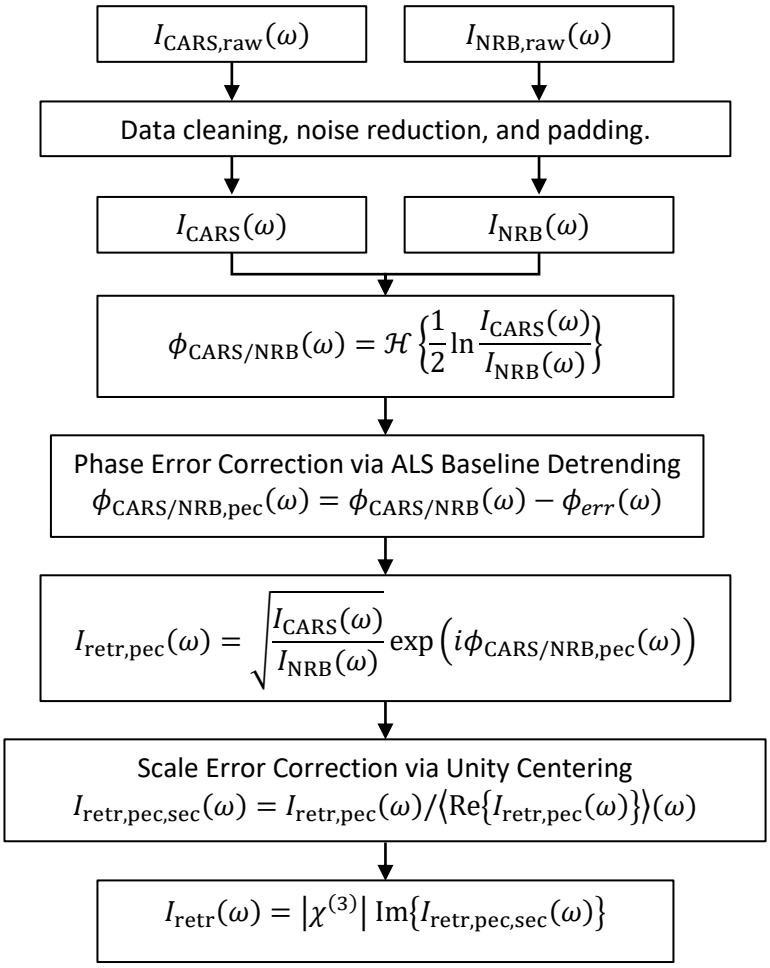

Figure 3: Flow diagram for the KK method. This diagram shows the procedure for implementing the KK method with phase and scale error corrections as described in Camp *et. al.* [12]. This is intended to provide the reader with a brief high-level overview of the KK method with error corrections without the derivation provided in the original source.

2.1.2 Learned Discrete Hilbert Transform

The *Learned Discrete Hilbert Transform* (LeDHT) method of spectral phase retrieval uses the same approach as previously described for the KK method, however the DHT is replaced with a "learned" transformation matrix that computes the Hilbert transform on discrete data without the errors associated with the DHT, particularly the errors at the endpoints of the spectral window [17]. This method was inspired by the fact that every discrete linear transformation can be represented by a transformation matrix that can be applied through matrix multiplication. Although DHT matrices are well known, they yield identical results to the Fourier-based DHT used in the KK method, thus offering no improvement in accuracy. The LeDHT method seeks solve this issue by using the ordinary least squares (OLS) optimization method to solve for the matrix **H** that minimizes the residual sum-of-squares between the true and predicted Hilbert transforms according to the following equation:

$$\mathbf{H} = \text{argmin}_{\mathbf{H}} \|G - F\mathbf{H}\|^2 \tag{15}$$

Where F is a matrix of training data and G is a matrix of the known Hilbert transforms of the corresponding data in F. F and G are both $M \times N$ matrices, where M is the number of input spectra and N is the length of each spectrum [17]. Once the optimal matrix **H** is found, the LeDHT method can then be applied with matrix multiplication as,

$$G_{exp} = F_{exp}\mathbf{H} \tag{16}$$

Where F_{exp} is the matrix of experimental input spectra and G_{exp} is the matrix retrieved Hilbert transforms for the spectra in F_{exp}. If the obtained matrix **H** is optimal, the LeDHT method promises more accurate phase retrieval than the KK method.

2.1.3 Convolutional Neural Network: SpecNet

In 2020, SpecNet was the first deep-learning-based solution to the NRB removal problem [18]. The goal of SpecNet is to simplify NRB removal by training a convolutional neural network (CNN) to retrieve the Raman-like spectral line shapes directly from the CARS spectra without the need for a separate NRB measurement. Figure 4 shows a schematic diagram of the CNN architecture that SpecNet is based on.

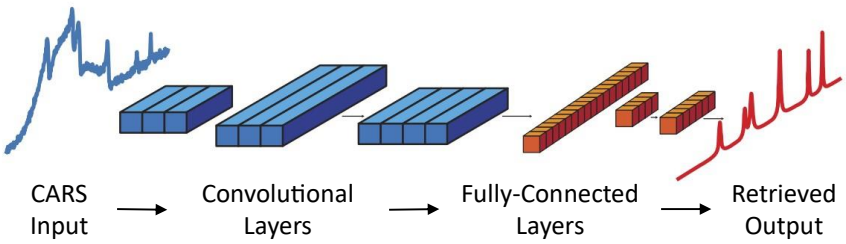

Figure 4: SpecNet schematic diagram. This figure shows a schematic diagram for the CNN-based architecture that the SpecNet model employs to solve the NRB removal problem. This figure is adapted from Valensise *et. al.* (2020). [18]

The SpecNet model maps the input CARS spectra to the target Raman spectra using a series of convolutional (CL) layers followed by fully-connected (FC) layers. The CL layers serve the purpose of learning progressively higher-order features of the input data by applying filters that select for the features that are relevant to the final retrieval. The FC layers then take the output of the CL layers and reconstruct the predicted Raman spectral line shapes from them. The CL layers additionally contribute to making the model translationally equivariant, meaning that the model learns each CARS feature in a way that preserves locality [19]. This is necessary to consider for spectral retrieval because the model should be able to retrieve a spectral feature in a way that preserves its location regardless of whether it has been shifted within the spectrum. Refer to Appendix C for a general

overview of neural networks and to Appendix D for specific details related to the implementation of the SpecNet model.

Due to the high generalizability of CNNs, the SpecNet model can be trained on simulated CARS spectra and then applied to experimental CARS spectra. The accuracy of the model, however, is contingent on how well the training data represents the real data. For this reason, the training dataset should be carefully designed to reflect the expected experimental spectra. Should the SpecNet model be trained properly, it promises to simplify the NRB removal process by making the requirement of an NRB measurement passe.

2.1.4 Convolutional Autoencoder: VECTOR

Following the creation of SpecNet in 2020, a search for more capable deep-learning models for NRB removal was commenced. Since convolution autoencoders (CAEs) are a natural successor to CNNs for this type of problem, a CAE-based approach called VECTOR was developed in 2022 [20]. A schematic depiction of the CAE architecture that VECTOR is based on can be found in Figure 5.

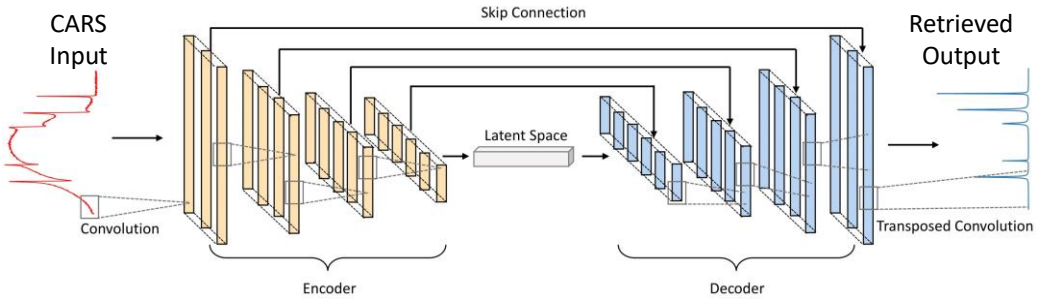

Figure 5: VECTOR schematic diagram. This figure shows a schematic diagram of the convolutional autoencoder architecture that the VECTOR model is based on. The model consists of an encoder and decoder pair joined by a latent space bottleneck. This figure is adapted from Wang *et. al.* (2022). [20]

At a basic level, autoencoders are comprised of two subnetworks: an encoder and a decoder. The encoder is trained to reduce the dimensionality of the input data into a so-called “latent space” representation that is presumed to be a superior encoding for the data, similarly to principal component analysis (PCA). However, the encoder is more flexible than PCA given that it is a non-linear transformation, whereas PCA is an orthogonal linear transformation. Using a latent space with finite number of nodes whose outputs are encoded by 32- or 64-bit binary values, our latent space can only represent a definite number of states. The goal of the encoder is to optimize the use of these states to maximally encode the features of the input CARS spectra that are relevant to the retrieval. The decoder then learns to reconstruct the target Raman-like line shapes from the encoder’s latent space representations of the input CARS spectra.

Convolutional autoencoders (CAEs) are simply autoencoders that employ convolutional layers in the encoder and transposed convolutional layers in the decoder. Hence, everything that was said about the convolutional layers for SpecNet also applies to VECTOR. There are two primary characteristics of CAEs that will be relevant to the later testing: noise reduction and overfitting. CAEs tend to reject noise since it does not resemble features that are relevant to the retrieval, e.g. the resonant CARS line shapes, so the encoder learns to suppress it. CAEs also tend to overfit to the training dataset, performing poorly on any data that deviates from it. This is due to the over-tuning the encoder-decoder networks to features specific to the training data. Take note of these two characteristics as they will be important later. An in-depth summary of each VECTOR model used throughout this work can be found in Appendix E.

2.2 CARS Simulation Methods

Only by directly comparing the retrieved and ground-truth spectra can we find the true capabilities of each method. Thus, the ground-truth Raman equivalencies must be known for the CARS spectra used during the analyses. It is difficult to obtain a reliable experimental surrogate for the ground-truth Raman spectrum corresponding to a given CARS spectrum, and any experimental dataset would necessarily be limited to only the materials available in the lab. These limitations are undesirable for the purposes of training the machine learning models, as we want the training dataset to be accurate and expose the models to a broad range of input data for the best generalization. Consequently, a training dataset consisting of pairs of idealized synthetic CARS and ground-truth Raman spectra are used for the training of the machine learning models, and a similarly constructed synthetic dataset is used for testing the NRB removal methods. These simulations allow for precise control over each aspect of the dataset. This ensures that a broad range of inputs are included in the training/testing datasets and that the results can be easily replicated by others. The following section outlines the procedure for generating the simulated data used herein.

2.2.1 Resonant Spectral Line Shapes

The resonant contribution to the third-order susceptibility is a complex-valued function that can be calculated explicitly for a given system using time-dependent perturbation theory [21]. For simplicity, the general solution for a system of damped harmonic oscillators, each having the form of a complex Lorentzian similar to that given by Lotem *et. al.* [3], will be used to simulate the resonant susceptibility in this work. The resonant

contribution to the third-order susceptibility is then given by the sum of these complex Lorentzian peaks as follows:

$$\chi_R^{(3)}(\omega) = \sum_{i=1}^{n} \frac{A_i}{\Omega_i - \omega - j\Gamma_i} \tag{17}$$

Where $\mathrm{A_i} \propto \sigma_\mathrm{i} \mathrm{C_i}$ is the amplitude proportional to the cross-section (σ_i) and concentration of scatterers (C_i), Ω_i is the central frequency, and Γ_i is the half-width of the i-th peak for a spectrum with n peaks. Figure 6 demonstrates an example of the real and imaginary parts of Eq. (17).

The parameters in the equation above are stochastically generated for each spectrum in the training and testing datasets according to the following conditions:

$$\begin{aligned} A_i &= \mathrm{U}(0.01,\ 1) \\ \Omega_i &= \mathrm{U}(300,\ 1900) \\ \Gamma_i &= \mathrm{U}(2,\ 30) \\ \mathrm{n} &= \mathrm{U}(1,\ 15) \in \mathbb{Z} \end{aligned} \tag{18}$$

Where $U(a, b)$ indicates that the variable is being chosen at random according to a uniform probability distribution between a (min) and b (max).

When calculating the Raman spectrum from $\chi_R^{(3)}$ above, we are inclined to adopt a normalization scheme such that: $0 < \mathrm{Im}(\chi_R) \leq 1$. By doing this we also ensure that the CARS spectrum is restricted to this range. Adopting such normalization schemes helps to avoid covariate shifts in the data that can negatively impact the performance of machine learning models [22].

Simulations of the spectra for real materials can be obtained by setting the peak amplitudes, centers, and half-widths according to the known values established by previous studies of those materials. These can be used for testing how well the methods work on idealized data for given materials, and they can be used to augment the training dataset for stronger results on experimental spectral retrieval of those materials.

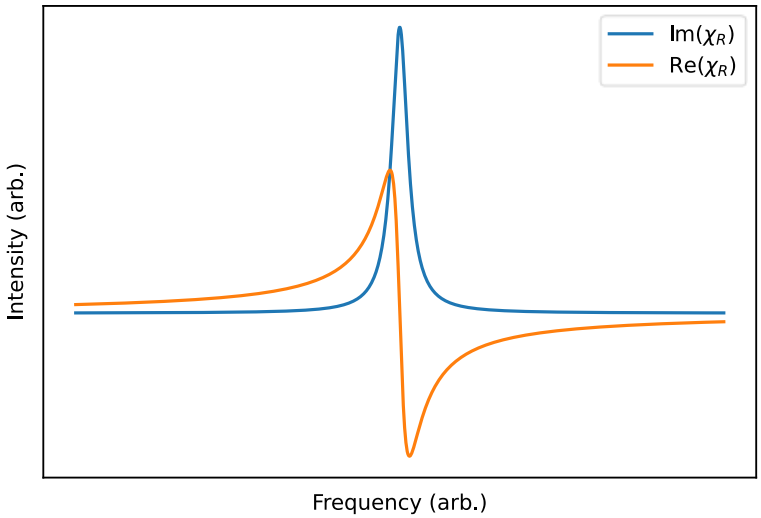

Figure 6: Resonant susceptibility example. This figure demonstrates the real and imaginary parts of the resonant susceptibility calculated with Eq. (17), with the imaginary part corresponding to Raman spectral line shapes and the real part contributing to the dispersive lines shapes in CARS spectra.

2.2.2 Non-resonant Spectral Line Shapes

The spectral profile of the NRB is system dependent and thus varies between measurements taken from different systems. This makes it difficult to simulate an NRB profile that is generally applicable to common experimental data. For consistency with previous literature on NRB removal [17], [18], [20], we adopt a simple NRB profile consisting of

the product between two oppositely facing sigmoidal functions. This idealized dual-sigmoid NRB can be described by the following equation:

$$\chi_{NR}^{(3)}(\omega) = \sigma(\omega, x_1, w_1)\,\sigma(\omega, x_2, -w_2) \tag{19}$$

Where σ is a sigmoid function given by,

$$\sigma(x, x_0, w) = \frac{1}{1 + e^{-(x-x_0)/w}} \tag{20}$$

Where x_0 represents the center position and w controls the effective width of the sigmoid. The NRB parameters are then stochastically generated such that:

$$\begin{aligned} w_1, w_2 &= \mathrm{U}(0.04\ \omega_{max}, 0.16\ \omega_{max}) \\ x_1 &= \mathcal{N}(0.3\ \omega_{max}, 0.2\ \omega_{max}) \\ x_2 &= \mathcal{N}(0.7\ \omega_{max}, 0.2\ \omega_{max}) \end{aligned} \tag{21}$$

Where $\mathcal{N}(\mu, \sigma)$ represents a normal distribution with a mean μ and a standard deviation σ.

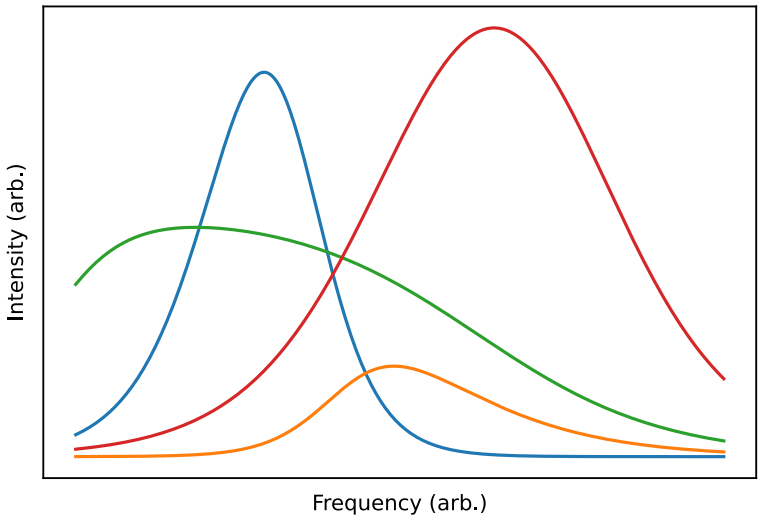

Figure 7: Non-resonant susceptibility examples. This figure shows several examples of the randomly generated non-resonant susceptibilities from the oppositely facing dual-sigmoid function described by Eq. (19).

2.2.3 CARS, NRB, and Raman Simulations

The total third-order susceptibility is calculated according to the above as follows:

$$\chi^{(3)}(\omega) = \alpha \frac{\chi_R^{(3)}(\omega)}{max\left(\left|\chi_R^{(3)}(\omega)\right|\right)} + \beta \frac{\chi_{NR}^{(3)}(\omega)}{max\left(\left|\chi_{NR}^{(3)}(\omega)\right|\right)} \quad (22)$$

Where each term in this equation is normalized, then the first term is multiplied by a factor α that controls the attenuation of the resonant component (i.e. the effective analyte concentration), and the second term is multiplied by the factor β representing the intensity of the non-resonant component. These factors are stochastically generated as follows,

$$\alpha, \beta = U(0.1, 1) \quad (23)$$

The intensity of the CARS signal is then given by,

$$I_{\mathrm{CARS}}(\omega) = \frac{1}{2}\left|\chi^{(3)}(\omega)\right|^2 \quad (24)$$

The intensity of the NRB signal is given by,

$$I_{\mathrm{NRB}}(\omega) = \frac{1}{2}\left|\chi_{NR}^{(3)}(\omega)\right|^2 \quad (25)$$

And the ground-truth Raman equivalency for the above is given by,

$$I_{\mathrm{Raman}}(\omega) = \mathrm{Im}\left(\chi_R^{(3)}(\omega)\right) \quad (26)$$

Where $\chi_R^{(3)}(\omega)$ in Eq. (24)–(26) is given by Eq. (22). Lastly, all spectra were simulated using 1000 data points from 0 cm^{-1} to 2000 cm^{-1} to cover the entire fingerprint region. Figure 8 demonstrates representative examples of the types of randomly generated spectra that are obtained from the equations described above.

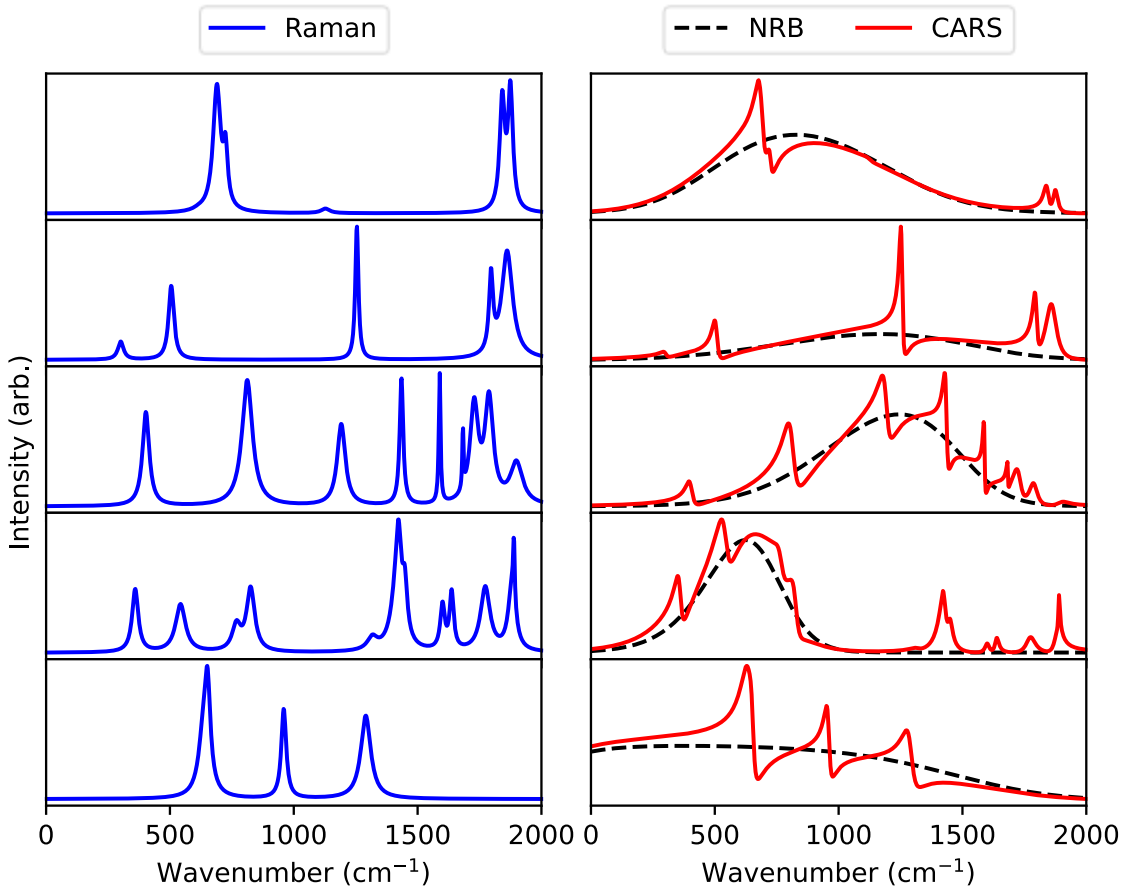

Figure 8: Randomly generated CARS, NRB, and Raman spectra. This figure shows 5 randomly generated spectra that are representative of the types of spectra that can be obtained using the methods described in Eq. (17)–(26).

2.2.4 Hyperspectral Image Simulations

To test the retrieval capabilities of the methods on hyperspectral images, the ground-truth hyperspectral Raman image must be known for comparison. We cannot obtain ground-truth references experimentally, so to accomplish this we generated synthetic hyperspectral images. To do this we can define the two-dimensional spatial distribution of each material in the form of a normalized raster image where each pixel intensity corresponds to the normalized material concentration at that location. Then the hyperspectral image is formed

by simulating the CARS and Raman spectra at each pixel according to the concentration encoded in the pixel intensity. This is described in detail below.

Consider that we want to simulate a hyperspectral image of a sample region containing M chemical species with various spatial distributions by generating a spectrum of length N for each pixel in an X × Y raster scan. To accomplish this, use the following procedure.

1. Create M normalized grayscale raster images to be used as intensity maps representing the effective spatial distributions of the M materials. These images can be represented by matrices $\mathbf{A}_m \in (0,1)^{X\times Y}$ for each $m \in [\![1, M]\!]$.
2. Simulate the resonant susceptibility $\chi_{R,m}(\omega) \in \mathbb{C}^N$ for each $m \in [\![1, M]\!]$, according to Eq. (17). These will be used to obtain the ground-truth Raman line shapes for each of the M chemical species.
3. Obtain the hyperspectral image containing the resonant susceptibility spectrum for each pixel by creating a data cube $\mathbf{R} \in \mathbb{C}^{\mathrm{X}\times\mathrm{Y}\times\mathrm{N}}$ where $\mathbf{R}_{xy} = \sum_{m=1}^{M} \mathbf{A}_{m,xy}\, \chi_{R,m}(\omega)$. Here we are multiplying by the coefficient representing the concentration of each material and then taking the sum of the susceptibilities for each material to represent the chemical mixing, for each pixel.
4. Simulate the non-resonant susceptibility $\chi_{NR}(\omega) \in \mathbb{R}^N$ according to Eq. (19). This will be used to obtain the NRB image $\mathbf{NR} \in \mathbb{R}^{X\times Y\times N}$ where $\mathbf{NR}_{xy} = \chi_{NR}(\omega)$.
5. Simulate the hyperspectral ground-truth Raman image, where $\mathbf{I}_{\mathrm{Raman}} = \mathrm{Im}(\mathbf{R})$.
6. Simulate the hyperspectral NRB image, where $\mathbf{I}_{NRB} = \frac{1}{2}|\mathbf{NR}|^2$.
7. Simulate the hyperspectral CARS image, where $\mathbf{I}_{CARS} = \frac{1}{2}|\mathbf{R} + \mathbf{NR}|^2$.

Note: the operations in steps 5-7 are applied elementwise.

2.3 Evaluation Methods

The mean absolute error (MAE), also referred to as the "L_1 norm", between the retrieved and ground-truth data is used to assess the accuracy of each method's spectral retrieval capabilities, where a lower MAE corresponds to a higher accuracy. The MAE is defined as follows:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|I_{\mathrm{retr},i} - I_{\mathrm{Raman},i}\right| \tag{27}$$

Where I_{retr} is the set of retrieved spectra, I_{Raman} is the set of ground-truth spectra, and N is the number of spectra over which the MAE is being calculated. Although a normalized accuracy metric like percent difference may be more interpretable than the MAE because it scales with the expected value, it is ill-behaved for values near zero. Since we expect to encounter values near zero in between the peaks in our spectra and for spectra with low intensities, the percentage difference should be avoided. Simply using the MAE is thus the most straightforward approach. Although we will not be able to ascertain a normalized percentage accuracy, we can still calculate the MAE for each method using the same data to compare their accuracies.

To evaluate the NRB removal methods on the simulated hyperspectral images, the retrieval methods were applied to the simulated hyperspectral CARS images to obtain the "retrieved" hyperspectral Raman images. These were compared to the ground-truth Raman images using the *structural similarity index measure* (SSIM), the preeminent metric in image processing for quantifying the similarity between two images [23]. The SSIM is defined as follows:

$$\mathrm{SSIM}(\boldsymbol{x},\boldsymbol{y}) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{28}$$

Where μ_x and μ_y are the pixel means and σ_x^2 and σ_y^2 are the pixel variances of images $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. σ_{xy} is the covariance of images $\boldsymbol{x}$ and $\boldsymbol{y}$. $c_1 = 0.0001$ and $c_2 = 0.0009$ are constants that ensure division stability when the denominator is near 0. SSIM exhibits properties of boundedness ($SSIM \leq 1$), and unique maximum ($SSIM(\boldsymbol{x},\boldsymbol{y}) = 1$ if and only if $\boldsymbol{x} = \boldsymbol{y}$), which are essential when quantitatively comparing two images.

2.4 Model Training Methods

Machine learning models must undergo a "training" process to be useful at a given task. This training process dictates how well the final model will perform, with the objective of finding the optimal performance for the given architecture. The models begin inept since they are initialized with random parameters, but the model parameters are gradually optimized for accomplishing the task through training. This process is where the "learning" originates in "machine learning". SpecNet and both VECTOR models were trained using an on-the-fly stochastically generated dataset of 25,600 spectra *without* additive noise for each of the 10 training epochs, an epoch being one iterative step through the training data. The default Adam optimizer [24] provided within the TensorFlow framework (version 2.13.0) was used during training with the mean squared error (MSE) as a loss function. Appendix F contains the Python code I adopted for training each model using the tf.keras.Model.fit method available in TensorFlow. This simplistic training regime was chosen with the intention that it could be easily adopted by those not experienced in machine learning to quickly train models on low-end hardware, such as a personal computer, for their personalized needs. Better results may be achievable by experienced

machine learning practitioners who implement customized training procedures on high-end purpose-built hardware. However, this simple training regime is sufficient to obtain acceptable results for the sake of evaluating the methods.

Additive noise was omitted from the training dataset since it slows down learning convergence and the convolutional layers in both architectures have internal noise reduction properties. This occurs because the convolution kernels that are learned by the convolutional layers during training filter the data at each layer according to the spectral features relevant to the retrieval. Noise does not resemble these features and so it is progressively filtered out by the convolutional layers. This will become apparent in Section 3.4 when we test the methods on noisy input data. Note that noise removal is not of principal concern in this thesis but is simply a property of the models. There are many dedicated noise removal techniques that are far more capable than the methods considered in this thesis and that should be applied to input data prior to NRB removal.

3 Results and Discussion

This section presents the results obtained in evaluating the spectral retrieval capabilities of the methods outlined in the previous section. These results include: the machine learning training results, the overall spectral retrieval accuracy, the hyperspectral image retrieval performance, adaptability to various noise and NRB conditions, and applicability to quantitative analysis. A brief discussion is provided for each of the results to add context and explain the significance.

3.1 Model Training Results

Figure 9 demonstrates the evolution of the mean absolute error (MAE) during training for SpecNet, VECTOR-8, and VECTOR-16. Evidently, each of the models converge within 10 epochs with a training MAE of ~ 0.015 for SpecNet and ~ 0.008 for the VECTOR models. The MAE is reduced to < 0.1 for each model within the first epoch, showing a marked improvement over the randomly initialized models created prior to training. Although SpecNet outperforms both VECTOR models during the first epoch due to its relative simplicity, the other two models quickly overtake SpecNet, as their complexity leads to greater generalization capabilities.

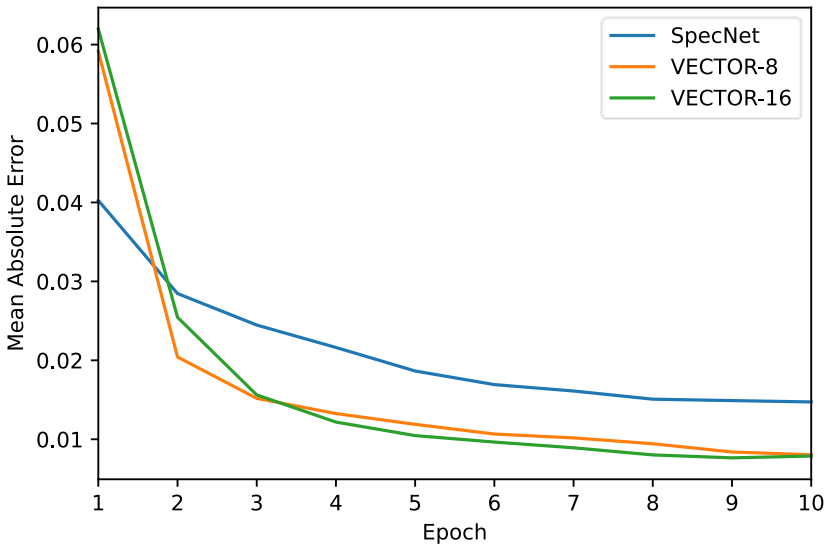

Figure 9: Training histories for three machine learning models. This figure shows the MAE progression during training for SpecNet, VECTOR-8, and VECTOR-16. Each model was trained with an on-the-fly stochastically generated dataset of 25,600 spectra for 10 epochs with a batch size of 256. The Adam optimizer was used with a MSE loss function.

Although the authors of the SpecNet model did not explicitly show their training curves, the results obtained later in this section are consistent with those presented in their article [18]. This demonstrates that the SpecNet model is reproducible, and that similar performance can be achieved under different training regimes. The authors of the VECTOR model did show their training results, and our results presented here are consistent with their findings to within the first 10 training epochs. However, the VECTOR authors trained their model using 100 epochs with variable learning rates and an optimizer based on traditional stochastic gradient descent. Although stochastic gradient descent converges slower than the Adam optimizer, it often produces a slightly superior model [20]. Consequently, their training procedure would take an order of magnitude longer than that used here and result in an MAE improvement of ~ 0.005 on average, which corresponds

to $< 1\%$ of the maximum peak amplitude. This will make a significant difference for exceedingly small peaks in an artificially clean spectra, but will fall below the noise in experimental data in most cases anyway. Thus, the much longer training time is unlikely worth the miniscule enhancement in performance.

3.2 Spectral Retrieval Comparison

A dataset of 10,000 random spectra was generated to test the efficacy of the NRB removal methods. Figure 10 presents the calculated MAE between the ground-truth Raman spectra and the retrieved spectra for each of the methods when applied to the test dataset. The methods can be ranked from worst to best performance as: KK, LeDHT, SpecNet, VECTOR-8, and VECTOR-16. We can see that there is a gradual reduction in the MAE as we move towards more complex models, with each of the deep-learning-based models outperforming the DHT-based methods. This demonstrates that machine learning is a powerful tool for NRB removal and has the generalization capabilities to be competitive with the DHT-based methods.

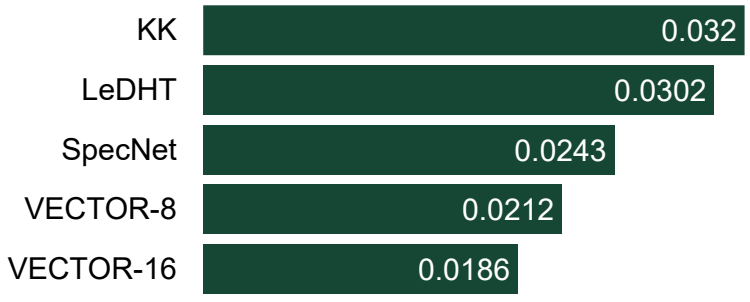

Figure 10: Comparison of MAE for NRB removal methods. This figure shows the MAE calculated between the ground-truth and retrieved spectra for each NRB removal method after being applied to a test dataset of 10,000 stochastically generated spectra. A lower MAE represents a better average accuracy.

The SpecNet and VECTOR models perform better than the DHT-based methods because they are optimized for the given retrieval, whereas the DHT and error-corrections both use computations that are prone to errors. There must then be a subset of spectra in the test dataset for which the KK and LeDHT methods perform suboptimally, such as those with flat NRB shapes. Assuming the machine learning models were trained on a dataset that sufficiently sampled the space of CARS spectra, the machine learning models must have been trained on the spectra for which the DHT-based methods perform poorly. The SpecNet and VECTOR models consequently have an advantage in that they do not rely on a predefined function to find the retrieved spectrum, and thus tend toward a more optimal solution. This demonstrates the power of feedforward neural networks as universal approximators [25], [26].

Although these MAE values can indicate how well the methods are retrieving the ground-truth spectra in general, they do not provide an indication of the circumstances under which each method performs best. As discussed in Section 2.3, the MAE also is not normalized by the spectral intensities. It simply shows the average absolute difference between the ground-truth and retrieved values. Consequently, it lacks the context that a percentage accuracy does. The MAE varies by about a factor of 2 between the best and worst of the methods, namely VECTOR-16 and KK, but without the context of how the MAE values relate to the spectral intensities it is hard to assess how these values pertain the performance in a meaningful sense. For example, this factor of 2 improvement in MAE equally applies to going from a percent difference of 80% to 40% or a percent difference of 2% to 1%. The latter of the two is clearly less significant in terms of the effect on the actual retrieval, as

anything below 10% may be considered sufficient. Further analyses will refine the testing criteria to probe the performance of each method under more controlled conditions.

3.3 Hyperspectral Image Retrieval

The previous section tested how well the methods performed on spectral retrieval in general. However, it is not obvious that this translates to the qualitative enhancement of the hyperspectral CARS images. To test their hyperspectral retrieval capabilities, we apply each method to simulated hyperspectral CARS images obtained as described in Section 2.2.4 and use the SSIM defined by Eq. (28) to see how well they retrieve the corresponding ground-truth hyperspectral Raman images.

Figure 11 shows the simulated ground-truth Raman and CARS spectra that were used to simulate the example **hyperspectral image**. The image chosen for the spatial intensity map of the simulated material is a microscopic fluorescence image of a tubulin-based microtubule network obtained from [27]. This image was chosen because it provides both fine and course details and a significant amount of contrast over the extent of the image. In particular, the image has a dark background that can be used to gauge how the chemical contrast compares to the NRB. The image resolution was scaled to 256 pixels on each size for a total of 65,536 square pixels, each representing a spectrum with 1000 points. In the figure below we see that the Raman image perfectly preserves the contrast of the original image (by definition), whereas the CARS image has decreased the contrast and effectively erased certain features of the image because of the NRB.

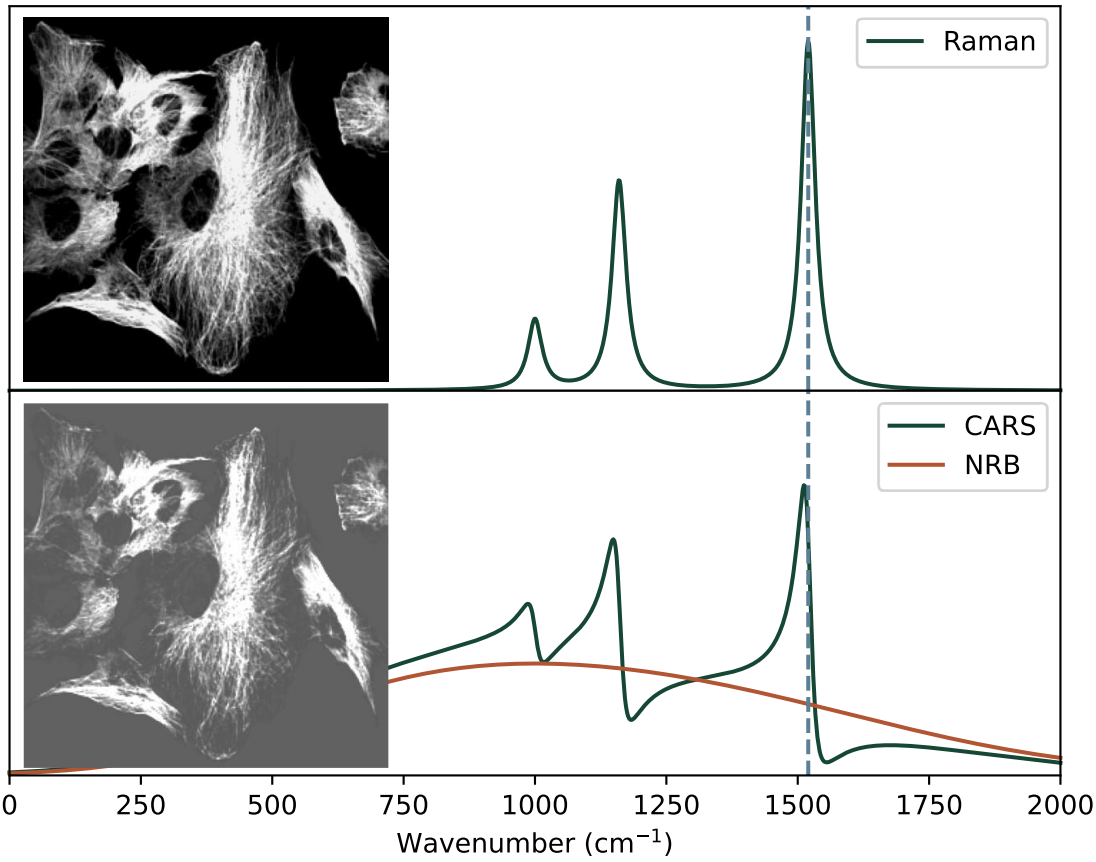

Figure 11: Simulated hyperspectral image example. This figure shows the simulated ground-truth Raman (top) and CARS (bottom) spectra used to simulate the material spectrum in the hyperspectral images. The inset images show the on-resonance frames indicated by the dashed blue line through the simulated Raman (top) and CARS (bottom) spectra. The inset images are adapted from a fluorescence image of a tubulin-based microtubule network retrieved from [27]. Note that the spectrum depicted in the figure does not represent the actual spectrum of tubulin.

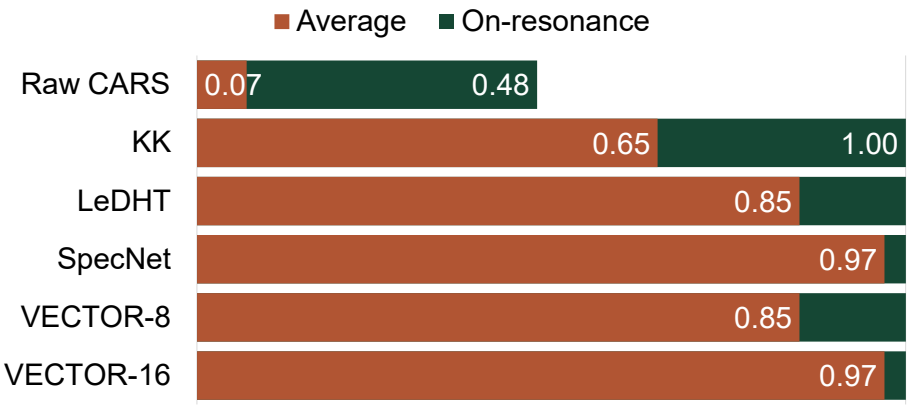

Figure 12: Average and on-resonance SSIM for retrieved and ground-truth images. The SSIM for the hyperspectral CARS image is included for comparison. The average SSIM values are calculated by taking the arithmetic mean of the SSIM between each corresponding image in the hyperspectral stack. The on-resonance SSIM is calculated just for one frame corresponding resonance peak from the spectrum in Figure 11.

Figure 12 above shows the average and on-resonance SSIM calculations between the retrieved and ground-truth hyperspectral images for each NRB removal method. The SSIM for the raw CARS image is also included in the evaluation. The CARS images have an SSIM of 0.07 on average and an SSIM of 0.48 on resonance. These are exceptionally low SSIM values that indicate that CARS images poorly represent the true chemical information and thus exemplify the need for spectral retrieval for reliable chemical analysis when implementing CARS microscopy.

We can see that all retrieval methods provide an improvement over the original CARS images, as indicated by the larger SSIM values. Peculiarly, the on-resonance SSIM for each retrieval method is 1, meaning that the retrieved images perfectly match the ground-truth images. The average SSIM is lower than 1 in all cases. This means that the retrieval methods perform better than average at or near the resonance peaks. This is particularly pronounced for the KK method, which has a much lower average SSIM than the other

methods, but still retrieves the on-resonance image perfectly. To explore this, the SSIM is plotted for each frame of the retrieved hyperspectral images (see Figure 13).

In Figure 13, we see a localized SSIM enhancement around the resonance peaks for the KK and LeDHT methods. This verifies that these methods perform better at or near the peak locations. Also, the KK and LeDHT methods were applied without padding or error-correction to elucidate their relative performance without external influence. The KK method thus performs very poorly at the endpoints of the data, as indicated by the SSIM dropping to zero at the ends of the plots. We can see that the LeDHT method successfully achieves its intended goal of fixing the endpoint errors associated with the KK method by increasing the SSIM at the endpoints [17].

SpecNet and the VECTOR models do not demonstrate the same localized performance enhancement near the resonance peaks as the KK and LeDHT methods do. Instead, SpecNet has random drops in performance that tend to be located towards the central region of the spectrum, which decreases the average performance. It is tempting to attribute this behaviour to the dropout layer, which randomly “drops out” certain nodes by setting their values to zero. However, this cannot be explained by the dropout layer in SpecNet (refer to Appendix D for SpecNet details) because the dropout layer is only applied during training according to the TensorFlow documentation [28]. Tracking down the source of this behaviour is thus nontrivial. Conversely, the VECTOR models perform well in the central region of the spectrum but have a decrease in performance towards the edges of the spectrum, reminiscent of the DHT-based methods. The performance improves when the depth of the model in increased, as evidenced by the increase in average SSIM from VECTOR-8 to VECTOR-16.

To summarize, the KK method does not appear to be the optimal solution under any circumstances. LeDHT and VECTOR-8 share the same average SSIM and have a similar SSIM spectrum, but LeDHT is preferable since the VECTOR-8 model produces an undesirable frame-to-frame variation that LeDHT does not. SpecNet and VECTOR-16 share an SSIM that is larger than the other models, making them preferable to the others. Although SpecNet has the same average SSIM as VECTOR-16, we see that the minimum SSIM of former is much lower than that of the latter, indicating that the worst-case retrieval for SpecNet is much worse than the worst-case retrieval for VECTOR-16. Since SpecNet exhibits less stability due to random dropouts in its SSIM, the VECTOR-16 model is the preferable for hyperspectral image retrieval according to these tests.

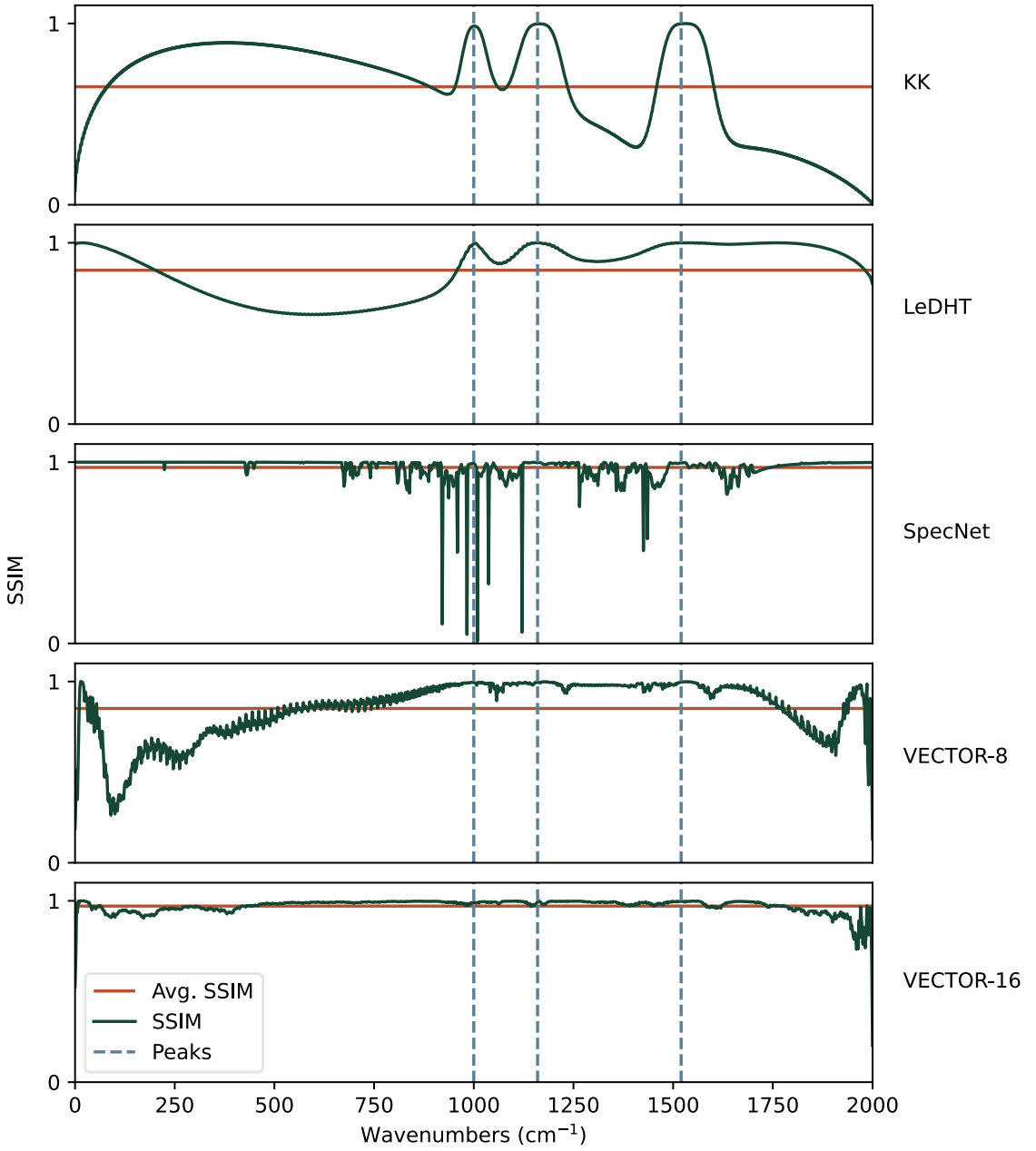

Figure 13: Retrieved SSIM for each frame in hyperspectral images. This figure shows the SSIM calculated for each frame in the pair of ground-truth and retrieved hyperspectral images. Blue dashed lines represent the resonance peaks in Figure 11. Note: the KK and LeDHT method were calculated without padding the input or error-correction calculations.

Table 1: Hyperspectral image retrieval times for each retrieval method.

Method	Retrieval Time (seconds)
KK	8
LeDHT	10
SpecNet	15
VECTOR-8	31
VECTOR-16	319

Previously, only the retrieval accuracy has been considered in the comparison between the spectral retrieval methods. However, the required compute time, versatility, and user-friendliness are also important to consider during the cost-benefit analysis. Table 1 shows the retrieval time required for the hyperspectral image from Figure 11. This is important for high-throughput and on-the-fly retrieval applications. Although the KK method is less accurate on average than every other method, it is the fastest method, taking 8 seconds per hyperspectral image or 0.1 milliseconds per pixel. Despite the VECTOR-16 model being the most accurate on average, it is significantly slower than the others, taking 319 seconds per hyperspectral image or 4.9 milliseconds per pixel. Moreover, the KK method can be used for spectra of any length, whereas LeDHT, SpecNet, and the VECTOR models all have fixed input sizes and need to be modified and retrained for each additional input spectrum length. This reduces the versatility and user-friendliness of the machine learning methods and increases the pre-use cost of implementing the machine learning methods. For this reason, the KK method is the most practical method despite its lower accuracy.

The times in Table 1 were achieved using a PC with 16 GB of RAM, an AMD Ryzen 5 3600 CPU, and an NVIDIA GeForce GTX 1660 Super GPU.

3.4 Impact of Noise and NRB Levels

To assess the robustness of each spectral retrieval method for experimental analysis, it is important to consider how each of the methods would perform when applied to input spectra with various signal-to-noise and signal-to-background ratios. For that purpose, this section will evaluate the spectral retrieval capabilities of each method when applied to input spectra with various noise levels and χ_R/χ_{NR} ratios. For the following comparisons, a spectrum with a single peak located at the center is used for the retrievals. Gaussian noise is added to the CARS and NRB spectra with a mean of zero and a standard deviation, referred to as the "noise level", that determines the amount of noise. Gaussian noise is chosen for simplicity and because it is commonly encountered in experimental data. No noise reduction techniques are applied to the input spectra to assess the true capabilities of each.

Figure 14 shows a plot of the input CARS spectrum for each combination of the following noise and NRB parameters: the rows represent the χ_R/χ_{NR} ratios that are varied between 100 to 0.01 by powers of 10 from top to bottom, and the columns represent the noise levels that are varied between 0.001 and 0.1 by powers of 10 from left to right. The top row represents CARS spectra for which the NRB has negligible effect on the peak shape. Each row moving down then increases the NRB in proportion to the resonant signal, making retrieval more necessary for Raman-like peak extraction. The bottom row represents spectra for which the resonance peak is significantly obscured by the NRB. Each column from left to right has an increasing noise level, representing a decreasing signal-to-noise ratio. This becomes significant for the CARS spectra in the bottom-right panel and the two adjacent panels for which the noise level is well above or near the peak intensity. Retrieval

is consequently expected to fail in these cases for each method without substantial noise reduction since the information about the resonance peak is completely buried.

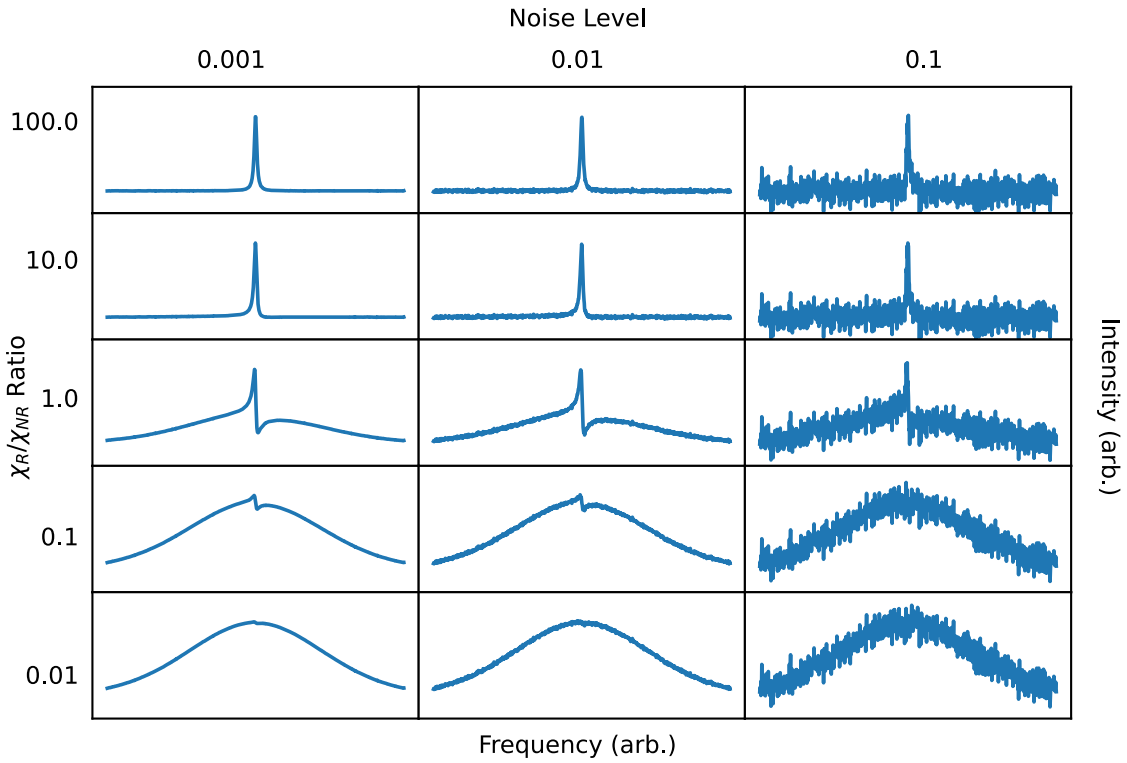

Figure 14: CARS test spectra with various noise and NRB parameters. The spectra are designed such that there is a broad symmetric NRB with a single centrally located resonance peak. The χ_R/χ_{NR} ratio is varied between 0.01 to 100 by powers of 10. The spectra were then normalized prior to adding Gaussian noise with a mean of zero and a standard deviation of 0.001, 0.01, and 0.1. Note: in the bottom-right spectrum and its two adjacent spectra, the peaks are of the same order as the noise level. It is thus not expected to retrieve these peaks without noise reduction techniques.

The following figures demonstrate the results of applying each spectral retrieval method to the spectra in Figure 14.

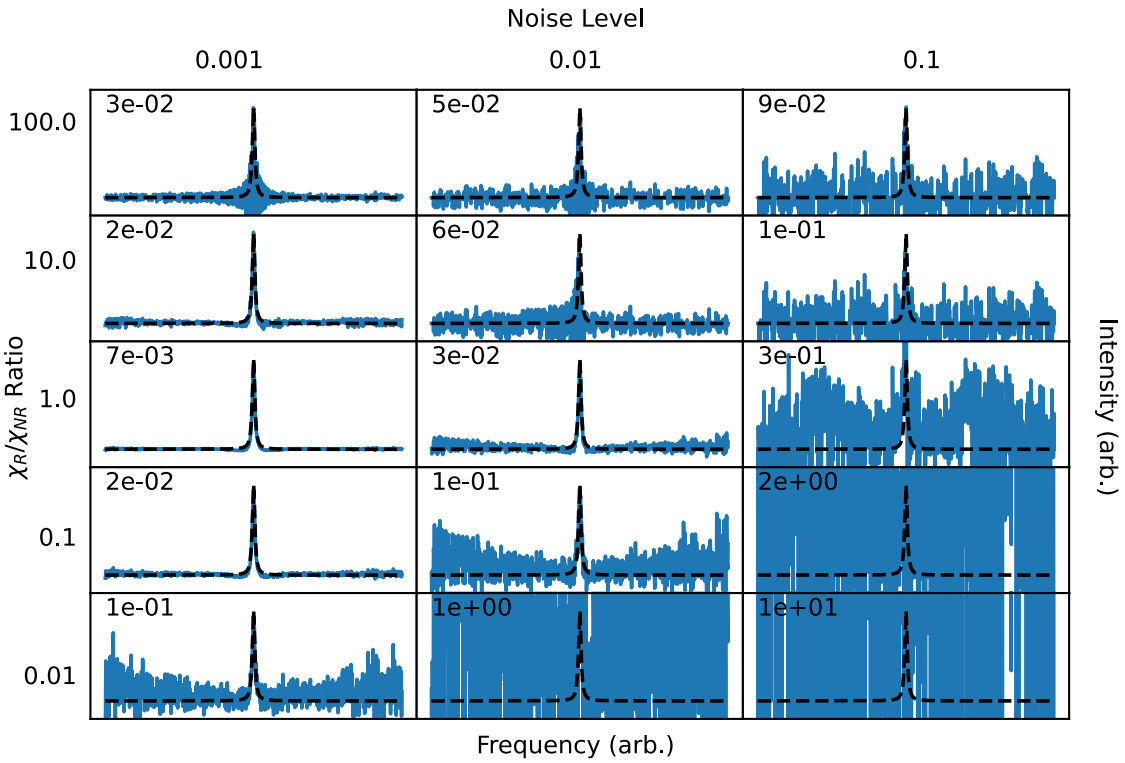

Figure 15: KK retrieval results for various noise and NRB parameters. This figure shows the retrieved spectra from applying the KK method to the CARS test spectra in Figure 14. Spectra retrieved from the The MAE normalized by the resonant peak intensity is shown in the top-left corner of each plot. Each dashed black line represents the ground-truth Raman-like spectrum to be retrieved.

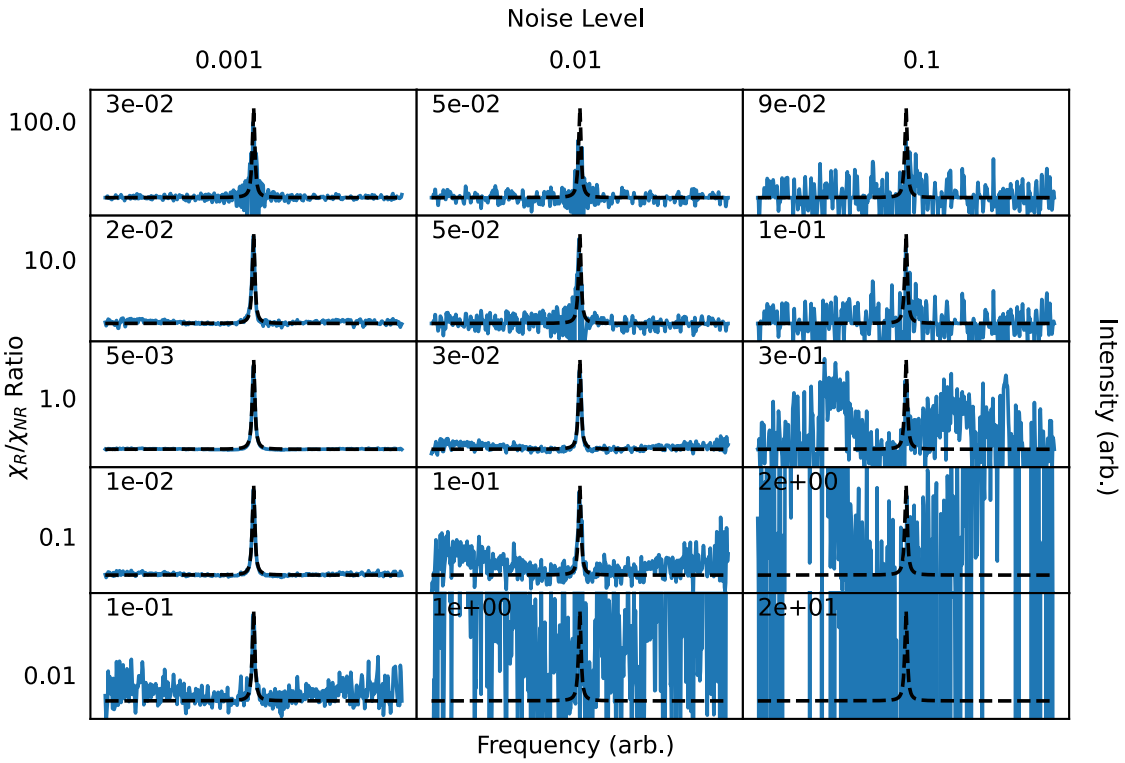

Figure 16: LeDHT retrieval results for various noise and NRB parameters. This figure shows the retrieved spectra from applying the LeDHT method to the CARS test spectra in Figure 14. The MAE normalized by the resonant peak intensity is shown in the top-left corner of each plot. Each dashed black line represents the ground-truth Raman-like spectrum to be retrieved.

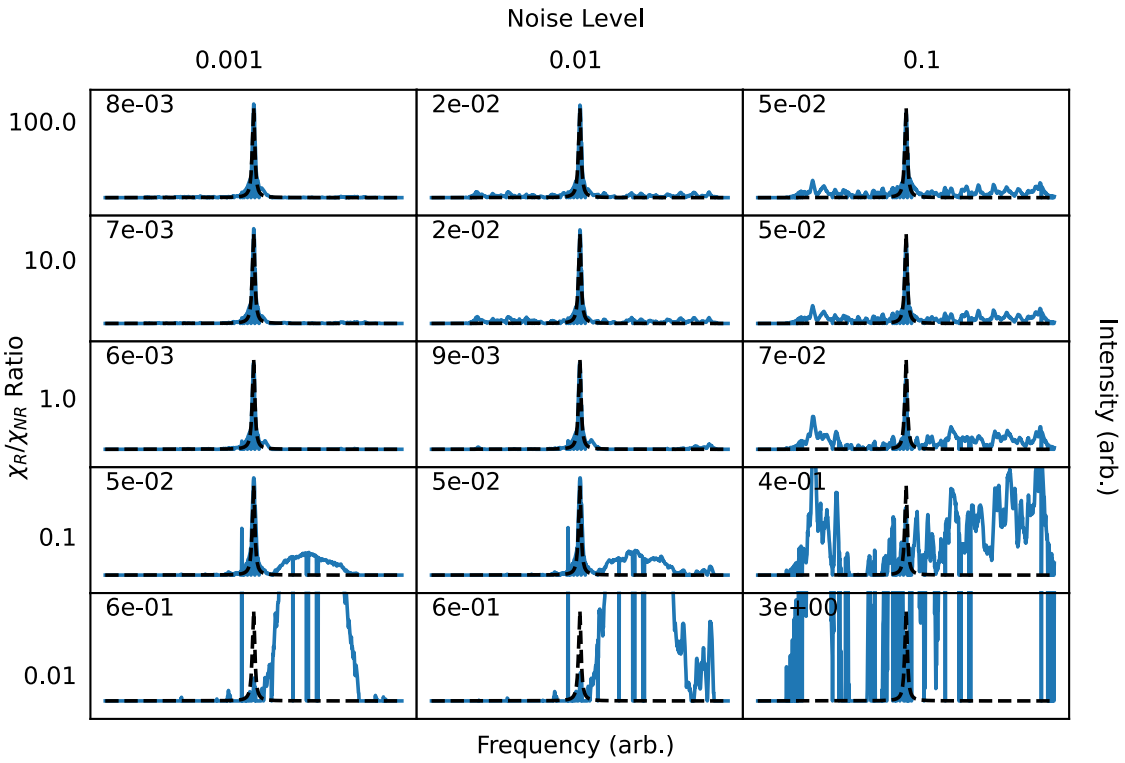

Figure 17: SpecNet retrieval results for various noise and NRB parameters. This figure shows the retrieved spectra from applying the SpecNet model to the CARS test spectra in Figure 14. The MAE normalized by the resonant peak intensity is shown in the top-left corner of each plot. Each dashed black line represents the ground-truth Raman-like spectrum to be retrieved.

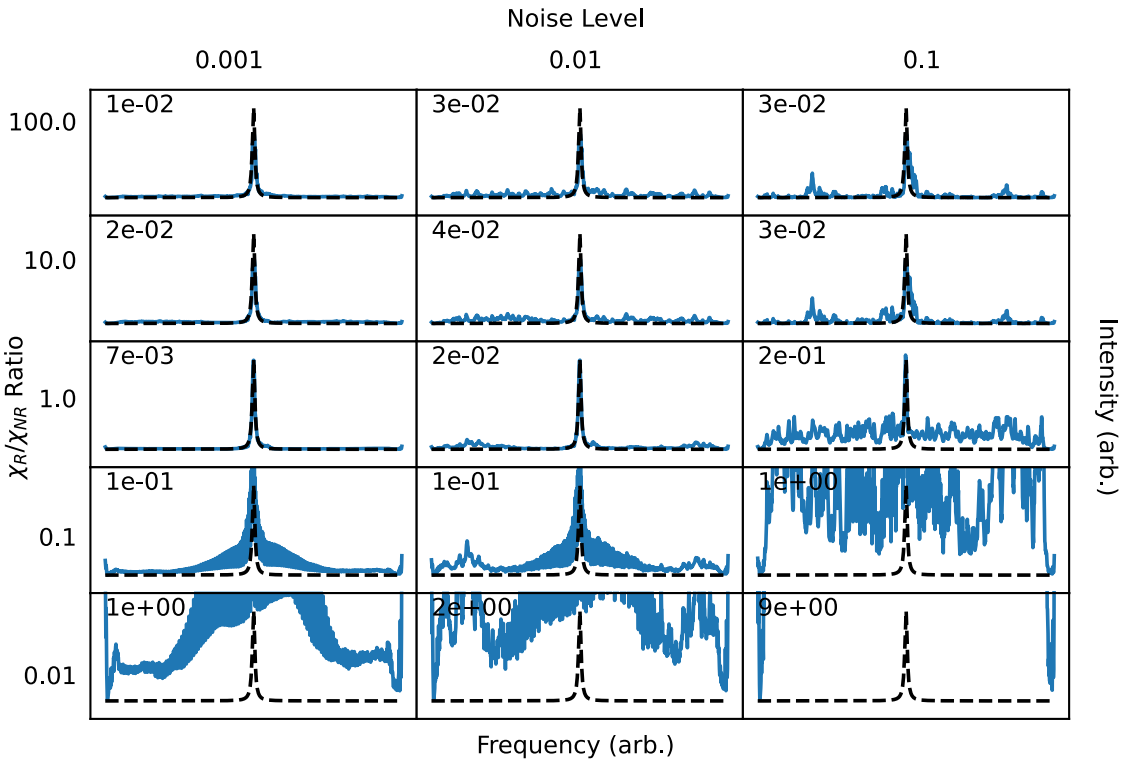

Figure 18: VECTOR-8 retrieval results for various noise and NRB parameters. This figure shows the retrieved spectra from applying the VECTOR-8 model to the CARS test spectra in Figure 14. The MAE normalized by the resonant peak intensity is shown in the top-left corner of each plot. Each dashed black line represents the ground-truth Raman-like spectrum to be retrieved.

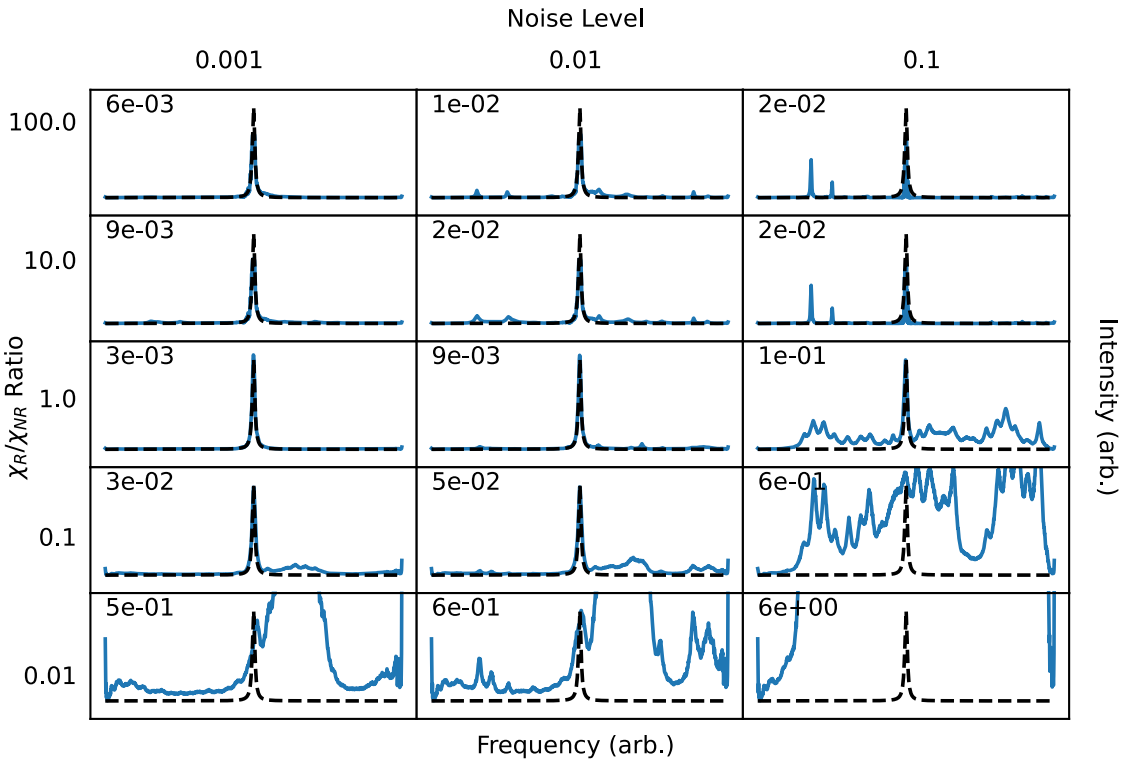

Figure 19: VECTOR-16 retrieval results for various noise and NRB parameters. This figure shows the retrieved spectra from applying the VECTOR-16 model to the CARS test spectra in Figure 14. The MAE normalized by the resonant peak intensity is shown in the top-left corner of each plot. Each dashed black line represents the ground-truth Raman-like spectrum to be retrieved.

3.4.1 KK Method: Noise and NRB

Since the discrete Hilbert transform (DHT) is a variance-preserving transformation, i.e. the variance of the DHT of a signal is the same as the variance of the signal itself [17], we expect noise to have a significant impact on the KK method. Typically, noise reduction techniques such as smoothing filters or singular value decomposition (SVD) are applied to the input spectra prior to using the KK method [29]. However, for this test we want to see the effect of the noise on the retrieval capabilities, so we do not perform any noise reduction.

Figure 15 shows the results of the KK method when applied to the spectra in Figure 14. We can see that the noise in the retrieved spectra increases along a given row from left to right, relating to the increasing noise level of the input spectra. This is consistent with our expectations given that the DHT preserves variance. Interestingly, the KK method also performs poorly when $\chi_R/\chi_{NR} = 100$ since the noise is amplified near the peak, degrading the reliability of the spectral retrieval near the resonance. This noise amplification is a consequence of dividing the CARS spectrum by a small NRB when the spectra are normalized to the unit interval, which effectively amplifies the noise. This may be alleviated by adopting a different standardized range for representing the data where a "small" NRB is still greater than 1. Thus, the CARS intensities are exceptionally large.

In summary, the KK method never performs well on data with a noise level of 0.1, and only performs well on data with a noise level of 0.01 if the signal-to-background ratio satisfies $0.1 < \chi_R/\chi_{NR} < 1$. The KK method is viable for data with a noise level of 0.001 as long as $\chi_R/\chi_{NR} \leq 10$.

3.4.2 LeDHT Method: Noise and NRB

Figure 16 shows the spectra retrieved from the LeDHT method when applied to the input spectra in Figure 14. As expected, the LeDHT method yields comparable results to that of the KK method. This is expected since the LeDHT method is identical to the KK method apart from the DHT calculation. Much of the discussion in the previous subsection about the KK method is applicable here as well, namely, the LeDHT method never performs well on data with a noise level of 0.1, and only performs well on data with a noise level of 0.01 if the signal-to-background ratio satisfies $0.1 < \chi_R/\chi_{NR} < 1$. The LeDHT method is also viable for data with a noise level of 0.001 as long as $\chi_R/\chi_{NR} \leq 10$.

3.4.3 SpecNet Model: Noise and NRB

Figure 17 shows the spectra retrieved from the SpecNet model when applied to the input spectra in Figure 14. It is immediately noticeable that the SpecNet model does not have the same issue with the high signal-to-background ($\chi_R/\chi_{NR} = 100$) spectra that the KK and LeDHT methods did. This is predominantly because the SpecNet model does not normalize the CARS signal by the NRB and was sufficiently trained to retrieve spectra with small NRBs. The SpecNet model does display the anomalous signal dropouts seen in the previous section, which points to a persistent error in the model architecture. Despite this, the SpecNet model does an exceptional job of retrieving the Raman-like peaks, so detecting the errors and interpolating between them may be an effective strategy to alleviate the issue.

We can see that that the SpecNet model also retrieves a spectrum with noise that increases from left to right. However, the SpecNet model does a much better job at suppressing the noise that the DHT-based methods owing to its convolutional layers. The SpecNet model most notably fails for spectra where $\chi_R/\chi_{NR} = 0.01$, representing an exceedingly small

resonance peak compared to the NRB. This is indicated by the failure of SpecNet retrieve the bottom-left peak. The SpecNet model also fails for the three peaks along the bottom-right corner with low signal-to-noise ratios, as expected. The SpecNet model does perform well in every other circumstance.

3.4.4 VECTOR Models: Noise and NRB

Figure 18 and Figure 19 shows the spectra retrieved form the VECTOR-8 and VECTOR-16 models, respectively, when they were each applied to the input spectra in Figure 14. Similarly to the SpecNet model, both VECTOR models solve the problem that the KK and LeDHT methods had with the $\chi_R/\chi_{NR} = 100$ spectra. Both VECTOR models also suppress the noise, however, VECTOR-16 does so better than VECTOR-8. The VECTOR-8 model demonstrates a sort of oscillatory artifact localized near the peak for noise levels of 0.001 and 0.01 when $\chi_R/\chi_{NR} = 0.1$, whereas VECTOR-16 does not. This type of behaviour is undesirable for spectral retrieval and thus gives VECTOR-16 the edge, as it performs well for all spectra except the three in the bottom right corner, as expected.

3.5 Quantitative Analysis

In experimental scenarios, we expect to encounter analyte concentrations that span several orders of magnitude. Since the resonant susceptibility scales linearly with analyte concentration, we can represent chemical concentrations of 100%, 10%, 1%, and 0.1% in our spectra by multiplying our resonant susceptibility by the factors 1.0, 0.1, 0.01, and 0.001, respectively. Since the NRB is in principle agnostic to the chemical concentration, we simulate the same generic NRB for all cases. A pure chemical sample represents an effective concentration of 100%, but even then the NRB is often substantial due to the relatively strong non-resonant four-wave mixing . So, we assume $\chi_R/\chi_{NR} \approx 1$ for each of the pure samples. For this test we will consider 5 simulated materials, each having a unique spectrum representing a different level of complexity, as shown in Figure 20. For the following quantitative analyses, 10 concentrations are considered for each material. Each concentration is given by a factor 10^x, where x is a set of 10 equally spaced values in the range $[-3, 0]$. Figure 21 shows the ground-truth Raman spectra for the materials at each concentration. The NRB removal methods are thus expected to retrieve each of these spectra during the analyses.

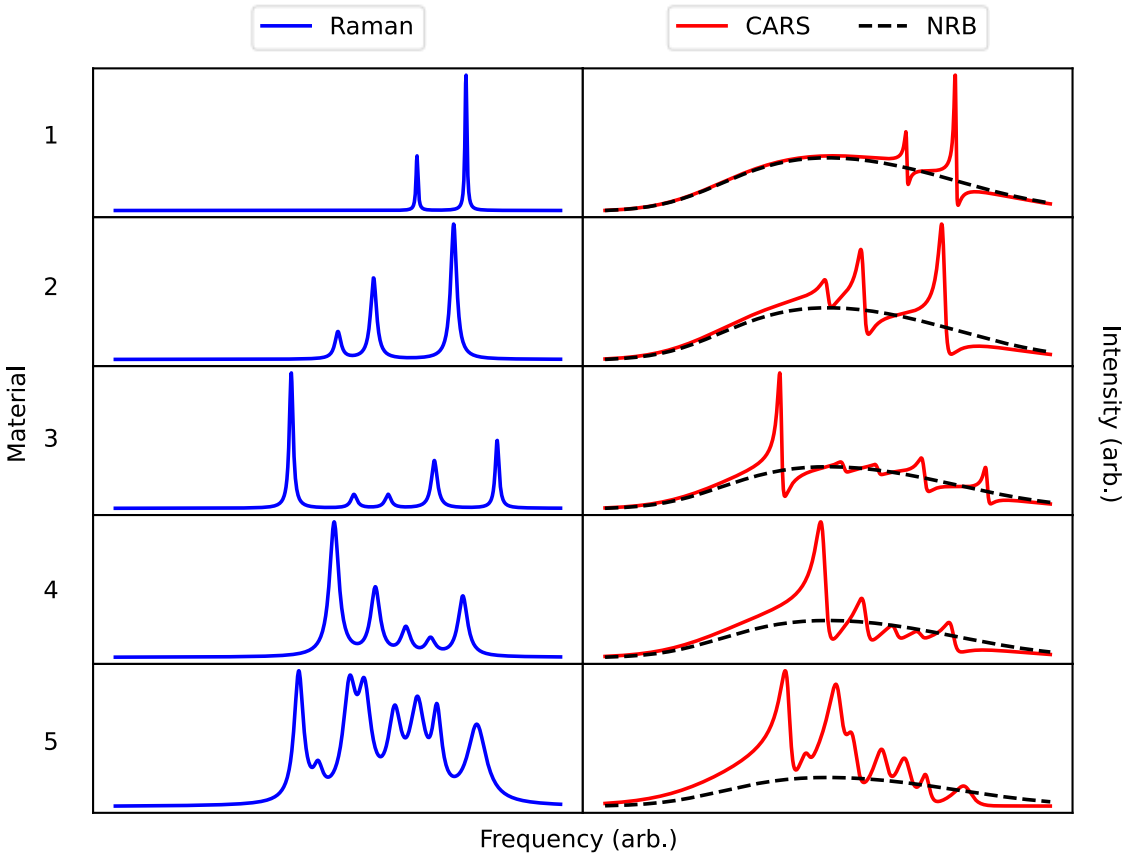

Figure 20: CARS spectra of simulated materials for quantitative analysis. The materials increase in complexity, with material 1 being the simplest and material 5 being the most complex. The complexity is qualified by the number and spacing of the peaks; with few distinct peaks being more simple and many convoluted peaks being more complex.

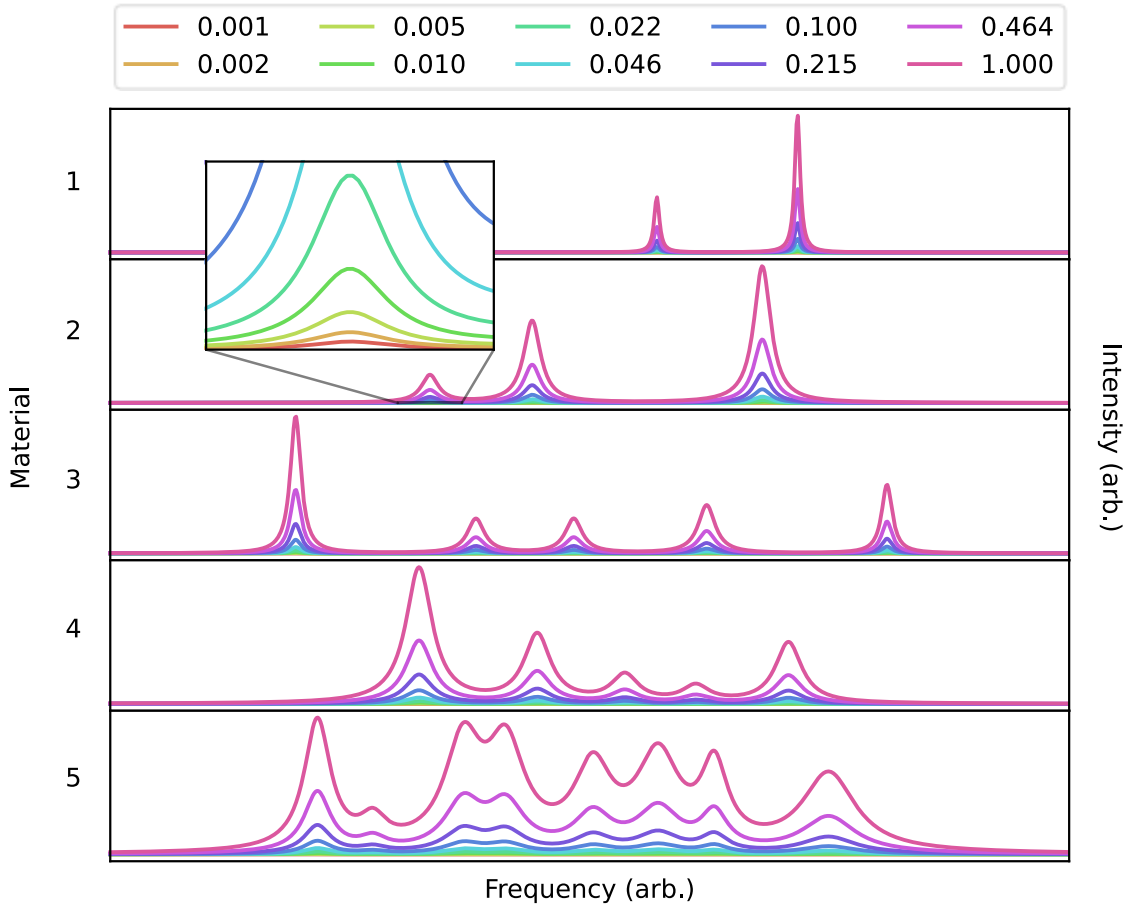

Figure 21: Ground-truth Raman-like spectra to be retrieved for quantitative analysis. This figure demonstrates the large variation in the ground-truth Raman intensities for the concentrations considered for quantitative analysis. The zoomed inset shows the lowest concentrations for the leftmost peak of material 2, which are not easily seen in the unscaled plots.

There are two ways in which one can determine concentration from Raman spectra for quantitative analysis: peak height or peak area. Using the peak height technique, the concentration is determined by locating a representative peak and determining its maximum intensity. Using the peak area technique, the concentration is determined by locating a representative peak and integrating over the extent of the peak to find its area. The peak area technique is more difficult to implement because it requires a reliable method

for estimating the extent of the peaks. For spectra with many convoluted peaks, this requires deconvoluting the spectra using a curve fitting optimization algorithm to find the individual peaks. These algorithms typically require special care for each spectrum to retrieve accurate results, and are thus difficult to implement in an automated fashion. For simplicity, the concentrations in the following analyses were found using the peak height technique by basing the concentration on the height of the largest peak. This requires only a peak-finding algorithm, which is reliable and proves to be highly accurate.

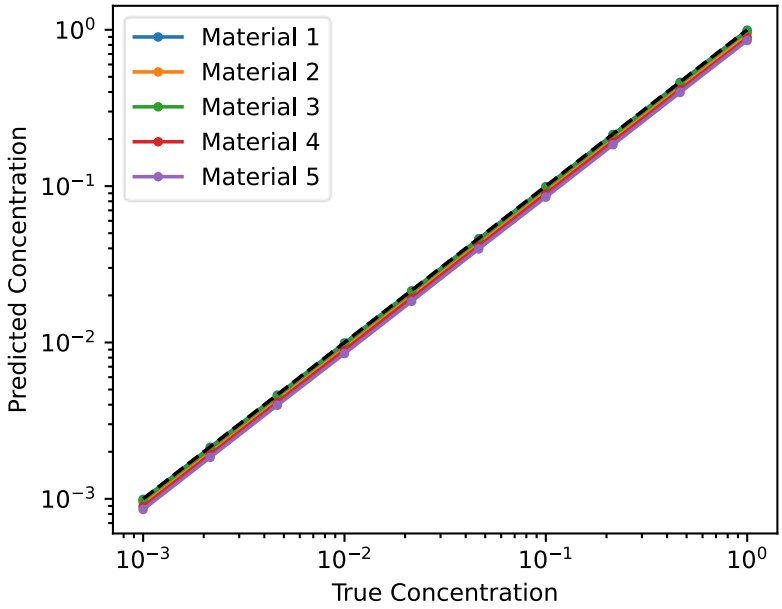

Figure 22: KK method quantitative analysis. This figure shows the concentration retrieved from the KK method (predicted concentration) versus the ground-truth concentration (true concentration) defined by the Raman spectra corresponding to the input CARS and NRB spectra. The black dashed line shows the perfect correlation.

3.5.1 KK Method: Quantitative Analysis

Figure 22 shows the results of the quantitative analysis using the KK method. We can see that the retrieved concentration is perfectly correlated to the ground-truth concentration for each material. However, the retrieved concentrations for materials 4 and 5 consistently underestimate the concentration. A likely explanation for this consistent error is that during phase error correction step the ASL baseline detrending algorithm uses a suboptimal smoothness parameter, resulting in the retrieved baseline fitting too closely to the peak, which subsequently removes a portion of the peak in the phase spectrum. This error then propagates through the retrieval. It is thus important to ensure that the optimal parameters are found when implementing the ALS baseline detrending algorithm for baseline removal.

Overall, the KK method demonstrates that it can be used to retrieve a consistent concentration over a broad range of concentrations as evidenced by the linear correlation between the predicted and true concentrations. This means that a concentration calibration curve can be created to account for the error mentioned above for each given material for consistent and comparable quantitative analysis.

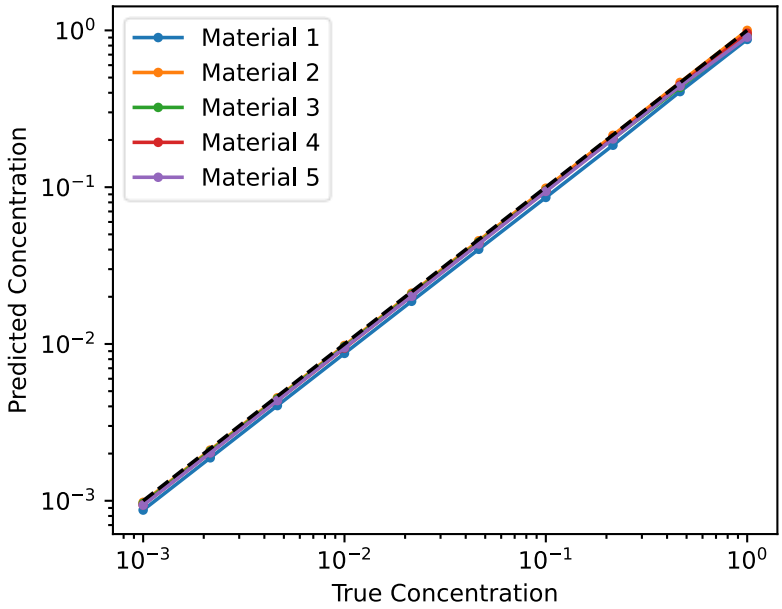

Figure 23: LeDHT method quantitative analysis. This figure shows the concentration retrieved from the LeDHT method (predicted concentration) versus the ground-truth concentration (true concentration) defined by the Raman spectra corresponding to the input CARS and NRB spectra. The black dashed line shows the perfect correlation.

3.5.2 LeDHT Method: Quantitative Analysis

Figure 23 shows the results for the quantitative analysis performed with the LeDHT method. The retrieved concentrations correlate to the true concentrations well. However, material 1 seems to show a consistent error that is much greater than material 4 or 5 in this case. This is a reflection of how the retrieved phase is calculated differently for the LeDHT method than the KK method, and so the consequent phase error correction and retrieval will yield slightly different results. Overall, this method demonstrates that it can predict the concentration with a high degree of consistency over a broad range of concentrations similarly to the KK method.

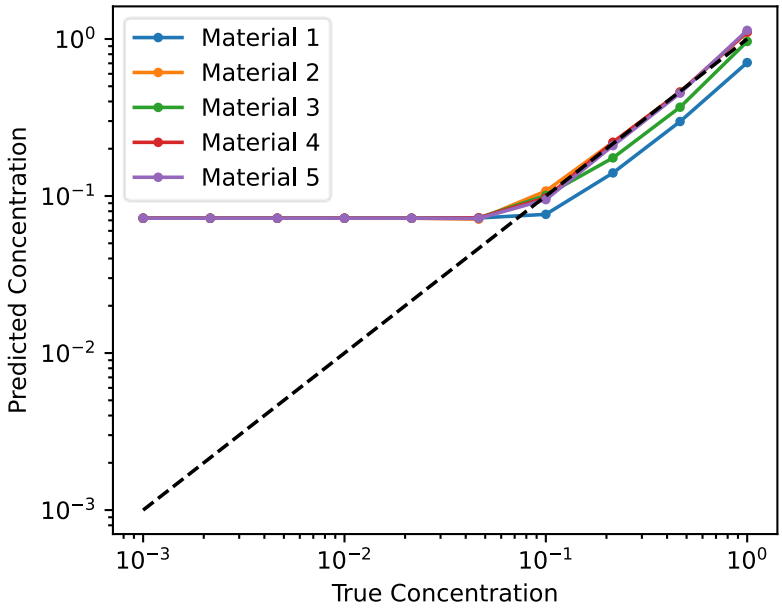

Figure 24: SpecNet quantitative analysis. This figure shows the predicted concentration retrieved from the SpecNet model versus the true concentration defined by the Raman spectra corresponding to the input CARS spectra. The black dashed line shows the perfect correlation.

3.5.3 SpecNet Model: Quantitative Analysis

Figure 24 shows the results of applying the SpecNet model to quantitative analysis. It is shown that the SpecNet model does not produce a strong linear correlation for a substantial portion of the testing range towards lower concentrations. Although there is a positive linear correlation for each material for concentrations from 0.1 to 1.0, the SpecNet model does not retrieve the correct concentrations below 0.1. Instead, the SpecNet model breaks down and retrieves a constant predicted concentration of ~0.07 for all concentrations below 0.1. This indicates that SpecNet is not a reliable method for quantitative analysis in general, and the concentration range should be carefully considered when using this model.

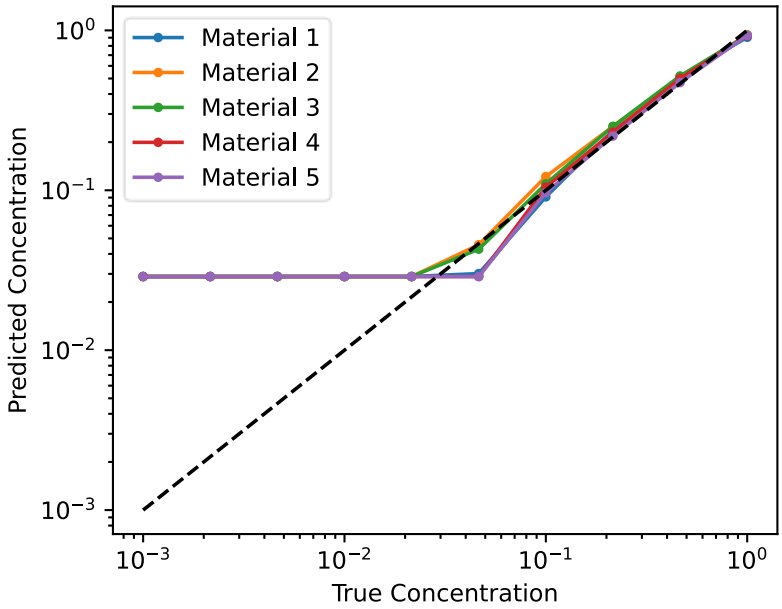

Figure 25: VECTOR-8 quantitative analysis. This figure shows the concentration retrieved from the VECTOR-8 model (predicted concentration) versus the ground-truth concentration (true concentration) defined by the Raman spectra corresponding to the input CARS spectra. The black dashed line shows the perfect correlation.

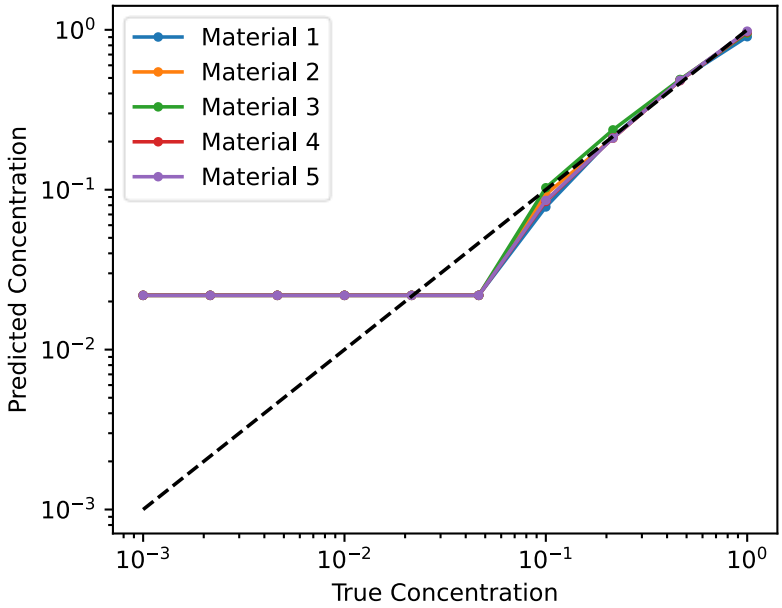

Figure 26: VECTOR-16 quantitative analysis. This figure shows the concentration retrieved from the VECTOR-16 model (predicted concentration) versus the ground-truth concentration (true concentration) defined by the Raman spectra corresponding to the input CARS spectra. The black dashed line shows the perfect correlation.

3.5.4 VECTOR Models: Quantitative Analysis

Figure 25 and Figure 26 shows the results for VECTOR-8 and VECTOR-16, respectively, when each were applied to the CARS spectra for quantitative analysis. Both models attain a linear correlation between the predicted and true concentrations for concentrations between 0.1 and 1.0. However, below 0.1 the predicted concentrations diverge from what is expected and settle on a constant value between 0.02 and 0.03. This indicates that the VECTOR models are slightly more generalized than the SpecNet model, but also limited by the restricted training dataset. The VECTOR models are thus not a reliable method for consistent comparable quantitative analysis unless trained on a highly tuned training dataset.

The deficient performance of SpecNet and the VECTOR models for low concentrations elucidates one of the major disadvantages of the machine learning approaches: they perform well only on data similar to their training data. The result of this is that the training data must be carefully designed to emulate every possible set of input data. Otherwise, the trained model will fail to work as intended for input data with parameters omitted from the training dataset. The training data used in this work considered effective concentrations down to 0.1. So, the models are expected to fail when the true concentration falls below that value. This can also be seen in Section 3.4 for each of the machine learning models, where they consistently fail to retrieve spectra for $\chi_R/\chi_{NR} = 0.01$, as that puts the effective concentration well below the minimum training concentration. This may be alleviated by retraining the models on an augmented training dataset to include lower χ_R/χ_{NR} ratios, at the expense of longer training times.

3.6 Experimental NRB Removal

The analyses of the NRB removal methods up to this point has been done using simulated data. This is fine for testing how the methods work in principle. However, the utility of the methods in practice can only be assessed by how they perform on experimental data. For this reason, it is necessary to extend the previous analyses to include experimental data. The CARS data used in a recent paper by Vernuccio *et. al.* [30] has been made available under Creative Commons Attribution 4.0 [31]. This acts as a free and openly accessible dataset which can be used for the experimental analysis.

Figure 27 shows a CARS spectrum of toluene included in the dataset [31]. This is a good reference spectrum for a comparison between the retrieval capabilities of each method since toluene has well-defined Raman peaks which we can compare to the retrieved spectra [32].

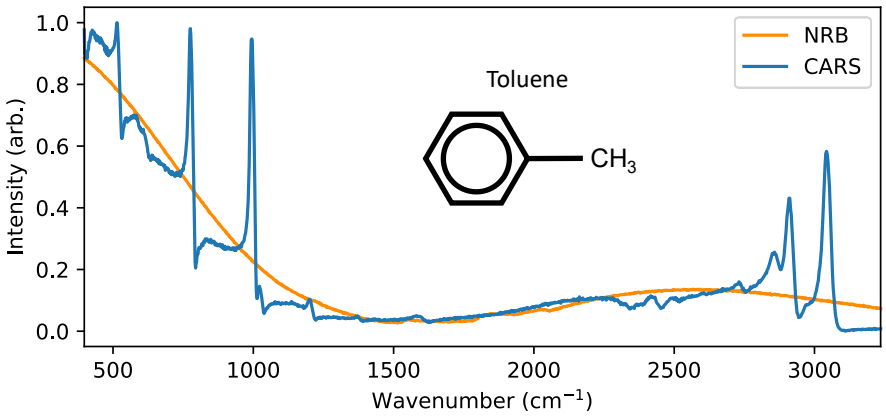

Figure 27: CARS spectrum of toluene. This figure shows the CARS spectrum of toluene with an inset image showing its skeletal structure. The data used here came from an open-access CARS dataset available under the Creative Commons Attribution courtesy of Vernuccio *et. al.* [31]. The NRB corresponding to the CARS spectrum was not provided with the open-access dataset, so we applied a Savitzky-Golay filter to the CARS spectrum to obtain an approximate NRB and Gaussian noise was added to simulate experimental noise.

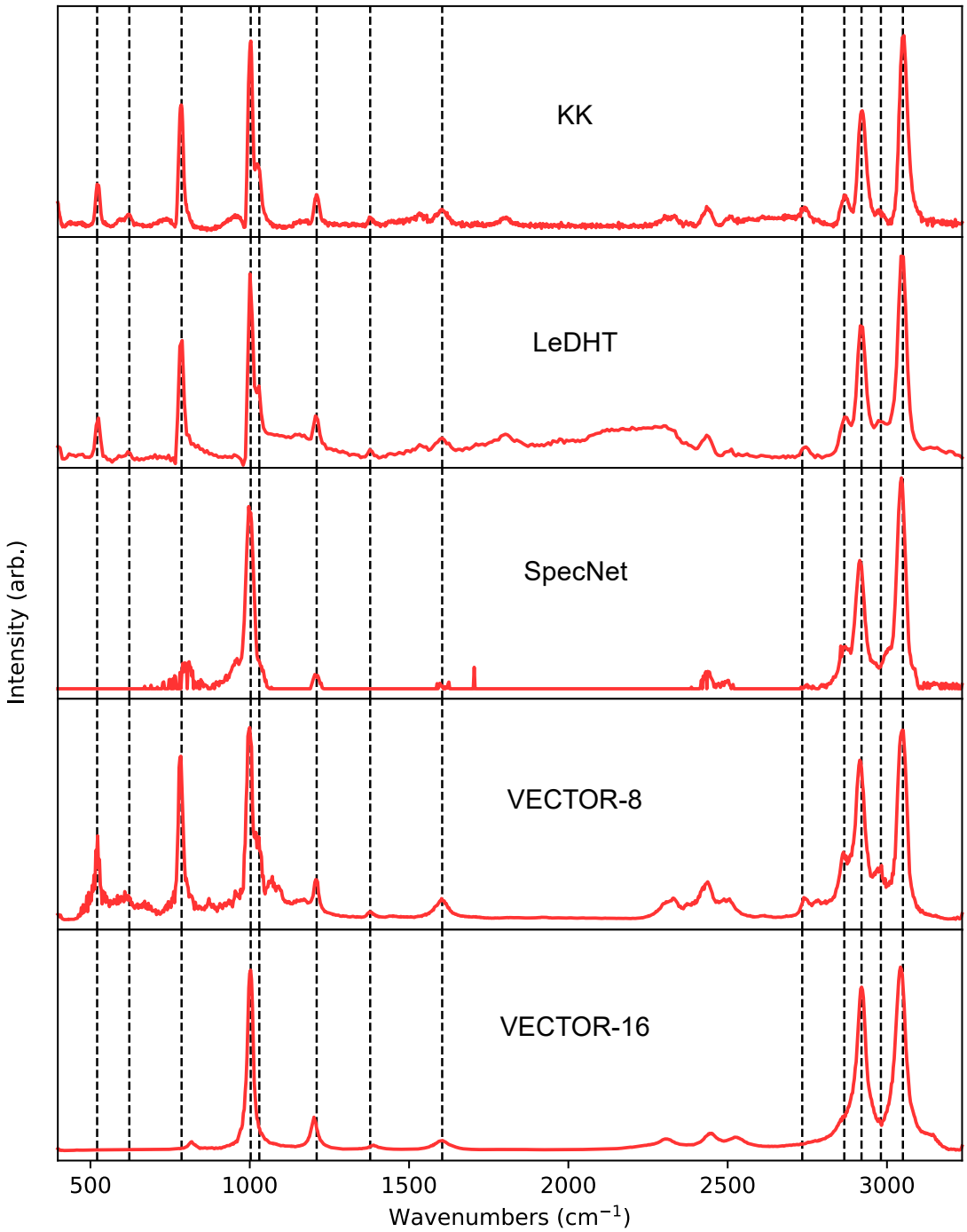

Figure 28: Retrieved spectra for toluene after NRB removal. This figure shows the spectra retrieved using the NRB removal methods on the CARS spectrum shown in Figure 27. The dashed black lines correspond to the known Raman resonances of toluene at wavenumbers of 521, 623, 786, 1004, 1030, 1208, 1379, 1604, 2736, 2870, 2920, 2983, and 3056 cm^{-1}, obtained from previous Raman studies [32]. The methods show a mixed ability to retrieve the peaks.

Figure 28 shows the retrieved spectra after applying each NRB removal method to the CARS spectrum of toluene in Figure 27. Each method demonstrates that it can remove the NRB, however they vary in their abilities to properly retrieve the Raman peaks. We can see that each method properly retrieves the primary CH peaks at 2920 cm^{-1} and 3056 cm^{-1}, as well as the fingerprint peaks at 1004 cm^{-1}, 1208 cm^{-1}, and 1604 cm^{-1}. However, the methods are more varied in their ability to pick up the less pronounced peaks. The KK, LeDHT, and VECTOR-8 methods all show a comparable ability to retrieve less pronounced peak. The SpecNet and VECTOR-16 models, however, both missed several of the peaks that the other methods properly retrieved. This may be evidence of training deficits, overfitting, or a lack of generalizability of the models. However, the machine learning models did suppress the noise from the experimental data very well, which may mean that more training on an augmented training dataset could improve performance.

Furthermore, VECTOR-16 retrieving a very smooth spectrum that is missing several of the peaks and the KK method retrieving a noisy spectrum that successfully retrieves those peaks indicates that a model that preserves the variance of the input data is less likely to erroneously dispose of relevant features of the data. This comes at the cost of noisier output data but will not smooth out real features. **In this sense, the analytical KK and LeDHT methods are preferable to the machine learning models.** It should be left to the end user to decide if a retrieved feature is a meaningful representation of a Raman peak or simply an artefact of noise or computational error, rather than delegating that decision to a machine learning model.

It is peculiar that VECTOR-8 demonstrates superior performance over VECTOR-16 when applied to the same experimental data. This indicates that there is a relationship between

model complexity and training resources in which a simpler model can reach superior performance when trained on less data. Hence, end users of machine learning models for NRB removal need to implement a model selection process that considers the performance of each model on their actual data prior to settling with a given model for the final analysis.

The dataset provided by Vernuccio *et. al.* also included hyperspectral CARS images of a liver sample which we can use to compare the contrast enhancement that each method affords after retrieval. This contrast enhancement is important for image processing procedures like segmentation, where regions of the image are separated into distinct chemical species. To compare the contrast enhancements from each method, we apply the retrieval methods on the hyperspectral CARS image to obtain the "retrieved" hyperspectral images, then compare the subjective contrast for on- and off-resonance frames. The off-resonance frames are taken from the so-called "silent" region (1800 cm^{-1} to 2800 cm^{-1}) where there are expected to be no vibrational resonances, and thus, no chemical contrast. The on-resonance frames are taken from the CH region (2800 cm^{-1} to 3100 cm^{-1}) where every organic molecule has broad resonance peaks in the Raman spectrum, and thus, there should be high chemical contrast between the background and chemical. The SSIM cannot be used in this case because the ground-truth image is not known for comparison.

Figure 29 (below) shows a comparison between the on- and off-resonance frames of the original hyperspectral CARS image and the retrieved images for each NRB removal method. We can see that each spectral retrieval method increases the on-resonance contrast while decreasing the off-resonance contrast. There are slight variations in the images that were retrieved by each, but since there are no ground-truth images to compare to, it is

difficult to tell which most accurately represents the true distribution of the material being imaged. Instead of comparing to a ground-truth image, we can assess the retrieved images based on their on their root mean square (RMS) contrast, which is essentially the standard deviation of pixel intensities. For the on-resonance frames, we can rank the retrieved images from best to worst as: VECTOR-8, KK, VECTOR-16, SpecNet, LeDHT.

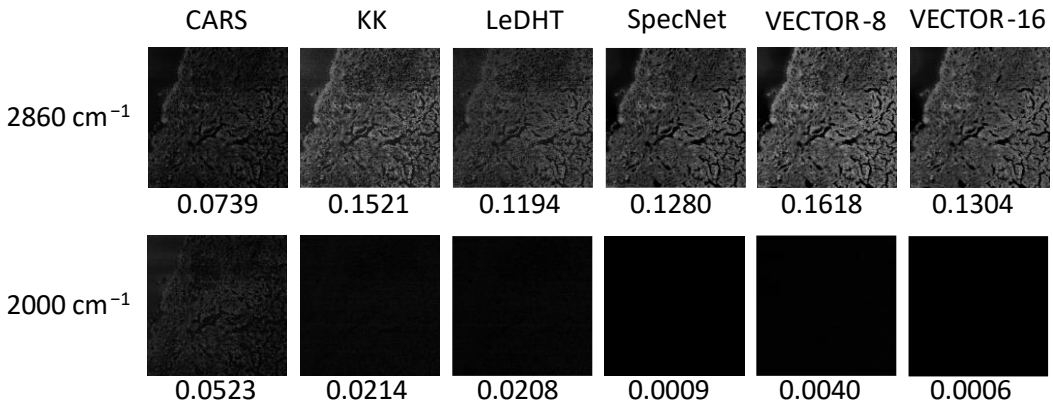

Figure 29: Comparison of chemical contrast after retrieval. This figure shows the chemical contrast of frames taken on-resonance (2860 cm^{-1}) and off-resonance (2000 cm^{-1}) from CARS and each NRB retrieval method. The off-resonance frame is chosen at 2000 cm^{-1} since that is in the so-called "silent" region where we do not expect any Raman peaks, whereas the on-resonance peak is taken at 2860 cm^{-1} corresponding to the CH region in which all organic materials have broad Raman peaks. The RMS contrast is labelled below each image. The data used here came from an open-access CARS dataset available under the Creative Commons Attribution courtesy of Vernuccio *et. al.* [31].

4 Concluding Remarks

The overarching goal of this thesis was to assess the efficacy of several spectral retrieval methods for CARS hypermicroscopy to identify which method is the best, and determine whether the NRB removal problem has been essentially solved. The spectral retrieval methods of interest were identified as the KK [9], [12] and LeDHT [17] methods, as well as the SpecNet [18] and VECTOR [20] models. These methods include both analytical and deep learning approaches to NRB removal. The methods were reproduced and trained (if applicable) independently, supporting the veracity and reproducibility of each of the methods as described in their respective articles. The NRB removal methods were then subjected to a systematic evaluation of the general spectral retrieval accuracy, hyperspectral image retrieval capabilities, robustness under various noise and NRB conditions, and whether the methods can be used for quantitative analysis. It was found that no single method performed better than the others in every test. For example, VECTOR-16 has the best overall spectral retrieval performance, however, when applied to experimental data it fails to retrieve peaks that the KK method successfully retrieves, despite the KK method having the worst general spectral retrieval performance. Nonetheless, the built-in noise reduction capabilities of the SpecNet and VECTOR models allow them to retrieve more accurately on noisy data than the DHT-based methods. The main conclusion from these tests is thus: results will vary. A method selection process is recommended prior to deciding on a given method for any application in order to obtain the best results. As for the ultimate question, has the NRB problem been essentially solved? Yes. Although minor improvements from better error-correction techniques or superior

machine learning models may be achieved in the future, the performance of each of the current methods is sufficient.

Bibliography

[1] E. Hecht, *Optics*, 5th ed. Pearson, 2016.

[2] R. W. Boyd, *Nonlinear Optics*, 4th ed. Academic Press, 2020.

[3] H. Lotem, R. T. Lynch, and N. Bloembergen, "Interference between Raman resonances in four-wave difference mixing," *Phys Rev A (Coll Park)*, vol. 14, no. 5, pp. 1748–1755, Nov. 1976, doi: 10.1103/PhysRevA.14.1748.

[4] J.-X. Cheng, A. Volkmer, and X. S. Xie, "Theoretical and experimental characterization of coherent anti-Stokes Raman scattering microscopy," *Journal of the Optical Society of America B*, vol. 19, no. 6, p. 1363, Jun. 2002, doi: 10.1364/JOSAB.19.001363.

[5] J.-X. Cheng, "Coherent Anti-Stokes Raman Scattering Microscopy," *Appl Spectrosc*, vol. 61, no. 9, pp. 197A-208A, Sep. 2007, doi: 10.1366/000370207781746044.

[6] J.-X. Cheng and X. S. Xie, "Coherent Anti-Stokes Raman Scattering Microscopy: Instrumentation, Theory, and Applications," *J Phys Chem B*, vol. 108, no. 3, pp. 827–840, Jan. 2004, doi: 10.1021/jp035693v.

[7] H. Pascher, "Stimulated Light Scattering in Solids," in *Encyclopedia of Materials: Science and Technology*, 2001. doi: 10.1016/b0-08-043152-6/01592-8.

[8] G. I. Petrov, R. Arora, V. V. Yakovlev, X. Wang, A. V. Sokolov, and M. O. Scully, "Comparison of coherent and spontaneous Raman microspectroscopies for

noninvasive detection of single bacterial endospores," *Proc Natl Acad Sci U S A*, vol. 104, no. 19, 2007, doi: 10.1073/pnas.0702107104.

[9] Y. Liu, Y. J. Lee, and M. T. Cicerone, "Broadband CARS spectral phase retrieval using a time-domain Kramers–Kronig transform," *Opt Lett*, vol. 34, no. 9, p. 1363, May 2009, doi: 10.1364/OL.34.001363.

[10] H. Kramers, "La diffusion de la lumière par les atomes," in *Atti del Congresso internazionale dei Fisici*, Como: Transactions of Volta Centenary Congress, Sep. 1927, pp. 545–557.

[11] R. de L. Kronig, "On the Theory of Dispersion of X-Rays," *J Opt Soc Am*, vol. 12, no. 6, 1926, doi: 10.1364/josa.12.000547.

[12] C. H. Camp, Y. J. Lee, and M. T. Cicerone, "Quantitative, comparable coherent anti-Stokes Raman scattering (CARS) spectroscopy: correcting errors in phase retrieval," *Journal of Raman Spectroscopy*, vol. 47, no. 4, pp. 408–415, Apr. 2016, doi: 10.1002/jrs.4824.

[13] P. H. C. Eilers and H. F. M. Boelens, "Baseline Correction with Asymmetric Least Squares Smoothing," *Life Sci*, 2005.

[14] Abraham. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures.," *Anal Chem*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964, doi: 10.1021/ac60214a047.

[15] M. H. Manghnani, A. Hushur, T. Sekine, J. Wu, J. F. Stebbins, and Q. Williams, "Raman, Brillouin, and nuclear magnetic resonance spectroscopic studies on

shocked borosilicate glass," *J Appl Phys*, vol. 109, no. 11, Jun. 2011, doi: 10.1063/1.3592346.

[16] G. R. Medders and F. Paesani, "Infrared and Raman Spectroscopy of Liquid Water through 'First-Principles' Many-Body Molecular Dynamics," *J Chem Theory Comput*, vol. 11, no. 3, pp. 1145–1154, Mar. 2015, doi: 10.1021/ct501131j.

[17] C. H. Camp, "Raman signal extraction from CARS spectra using a learned-matrix representation of the discrete Hilbert transform," *Opt Express*, vol. 30, no. 15, p. 26057, Jul. 2022, doi: 10.1364/OE.460543.

[18] C. M. Valensise, A. Giuseppi, F. Vernuccio, A. De la Cadena, G. Cerullo, and D. Polli, "Removing non-resonant background from CARS spectra via deep learning," *APL Photonics*, vol. 5, no. 6, p. 061305, Jun. 2020, doi: 10.1063/5.0007821.

[19] T. S. Cohen, M. Geiger, and M. Weiler, "A general theory of equivariant CNNs on homogeneous spaces," in *Advances in Neural Information Processing Systems*, 2019.

[20] Z. Wang, K. O' Dwyer, R. Muddiman, T. Ward, C. H. Camp, and B. M. Hennelly, "VECTOR: Very deep convolutional autoencoders for non-resonant background removal in broadband coherent anti-Stokes Raman scattering," *Journal of Raman Spectroscopy*, vol. 53, no. 6, pp. 1081–1093, Jun. 2022, doi: 10.1002/jrs.6335.

[21] G. L. Eesley, "Coherent raman spectroscopy," *J Quant Spectrosc Radiat Transf*, vol. 22, no. 6, pp. 507–576, Dec. 1979, doi: 10.1016/0022-4073(79)90045-1.

[22] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” Feb. 2015.

[23] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, IEEE, pp. 1398–1402. doi: 10.1109/ACSSC.2003.1292216.

[24] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[25] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, 1989, doi: 10.1016/0893-6080(89)90020-8.

[26] B. C. Csaji, “Approximation with Artificial Neural Networks,” 2001.

[27] P. Colarusso and C. Brideau, “Photons to Pixels: Illuminating Bit Depth,” *Bliq Photonics*, Aug. 03, 2022. https://bliqphotonics.com/what-is-bit-depth-in-microscopy/ (accessed Jul. 26, 2023).

[28] “TensorFlow v2.13.0 API Documentation: Dropout Layer.” https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout (accessed Aug. 11, 2023).

[29] C. H. Camp Jr *et al.*, “High-speed coherent Raman fingerprint imaging of biological tissues,” *Nat Photonics*, vol. 8, no. 8, pp. 627–634, Aug. 2014, doi: 10.1038/nphoton.2014.145.

[30] F. Vernuccio *et al.*, “Full-Spectrum CARS Microscopy of Cells and Tissues with Ultrashort White-Light Continuum Pulses,” *J Phys Chem B*, vol. 127, no. 21, pp. 4733–4745, Jun. 2023, doi: 10.1021/acs.jpcb.3c01443.

[31] F. Vernuccio *et al.*, “Supplementary material - Full-spectrum CARS Microscopy Of Cells And Tissues With Ultrashort White-light Continuum Pulses,” *Zenodo*, Apr. 20, 2023.

[32] J. K. Wilmshurst and H. J. Bernstein, “THE INFRARED AND RAMAN SPECTRA OF TOLUENE, TOLUENE-α- d_3, *m*-XYLENE, AND *m*-XYLENE-αα′- d_6,” *Can J Chem*, vol. 35, no. 8, pp. 911–925, Aug. 1957, doi: 10.1139/v57-123.

[33] S.-J. Baek, A. Park, Y.-J. Ahn, and J. Choo, “Baseline correction using asymmetrically reweighted penalized least squares smoothing,” *Analyst*, vol. 140, no. 1, pp. 250–257, 2015, doi: 10.1039/C4AN01061B.

[34] V. Maiorov and A. Pinkus, “Lower bounds for approximation by MLP neural networks,” *Neurocomputing*, vol. 25, no. 1–3, pp. 81–91, Apr. 1999, doi: 10.1016/S0925-2312(98)00111-8.

Appendix A: KK Method Details

A complete implementation of the KK method is available in the CRIkit2 software developed by Charles Camp Jr. (https://github.com/CCampJr/CRIkit2). Below are several simplified code snippets which are intended to help the reader understand how the KK method can be implemented in a small Python program.

```
import numpy as np
from scipy.signal import hilbert

def kk(cars, nrb):
    ''' Original implementation of KK phase retrieval method. '''
    F1 = np.log(np.sqrt(2*cars))
    F2 = np.log(np.sqrt(2*nrb))
    f1 = ifft(F1)[:len(F1)//2] # t > 0
    f2 = ifft(F2)[len(F2)//2:] # t < 0
    eta = np.concatenate([f2, f1])
    return -2*np.imag(fft(fftshift(eta)) - F1/2)

def dht(cars, nrb):
    ''' DHT-based implementation of KK phase retrieval method. '''
    return hilbert(0.5*np.log(cars/nrb)).imag
```

Each of the above functions return the phase given the CARS and NRB input spectra. This phase must then be corrected with the following baseline detrending algorithm. Note that the CARS and NRB spectra should be cleaned of any spurious values, noise reduced, and padded so that the DHT errors can be minimized speed up phase error correction.

The following code can be used to obtain the baseline of the phase spectrum, representing the phase error. This baseline can then be subtracted from the phase spectrum to retrieve the error-corrected phase spectrum. The code is based on the asymmetric least squares (ALS) smoothing algorithm proposed by Eilers and Boelens in 2005 [13].

```
import numpy as np
from scipy import sparse
from numpy.linalg import norm
from scipy.sparse.linalg import spsolve


def baseline_als(y, lam=1E3, p=1E-6, niter=10):
    L = len(y)
    D = sparse.diags([1,-2,1],[0,-1,-2], shape=(L,L-2))
    D = lam * D.dot(D.transpose())
    w = np.ones(L)
    W = sparse.spdiags(w, 0, L, L)
    for i in range(niter):
        W.setdiag(w)
        Z = W + D
        z = spsolve(Z, w*y)
        w = p * (y > z) + (1-p) * (y < z)
    return z
```

A more recent method called asymmetrically reweighted penalized least squares (arPLS) smoothing appears to have better performance and may be used in the future [33].

Appendix B: LeDHT Method Details

A complete implementation of the LeDHT method is available through the Hilbert toolkit software developed by Charles Camp Jr. (https://github.com/usnistgov/Hilbert). In this method, the optimal transformation matrix **H** described in Eq. (15) can be found by training on a dataset of pairs of Gaussian/Dawson or Lorentzian/dispersive line shapes. The matrix **H** used throughout this work was obtained from training on pairs of Gaussian/Dawsons line shapes even though the spectra were simulated with Lorentzian/dispersive line shapes. This is not significant since the transformation matrix trained on one is applicable to the other. As shown in the supplementary information of the

LeDHT paper, the Gaussian/Dawson-trained LeDHT matrix performs just as well on the Lorentzian/dispersive line shapes.

Appendix C: Artificial Neural Networks

At a basic level, artificial neural networks are nonlinear mathematical models that define mappings between inputs $\vec{x}$ and outputs $\vec{y}$. Each network is constructed by combining elementary units, referred to as nodes or neurons, in parallel into layers which are then combined layer-to-layer in series to support a feedforward computation where the output of each layer is the input to the next. The first layer is referred to as the input layer whose neurons take as their input the elements of the input vector, the last layer is referred to as the output layer whose neurons output the elements of the output vector, and the in-between layers are referred to as hidden layers. Each node/neuron in the hidden layers take as their inputs a weighted average of the outputs of each node/neuron in the previous layer, then apply a nonlinear activation function. Common activation functions include the rectified linear unit (ReLu), sigmoid, and tanh functions. Artificial neural networks "learn" through a process called *backpropagation* that optimizes the weights and biases used to calculate the weighted average in each neuron by minimizing an objective cost/loss function using an optimization algorithm referred as an optimizer. Optimizers typically utilize the concept of stochastic gradient descent and facilitate the backpropagation of errors. The mean squared error (MSE) or mean absolute error (MAE) between the target and predicted outputs are typically used as loss functions to be minimized, although any function can be used. The universal approximation theorem states that feedforward neural network are universal approximators [25], [26], [34], which has profound implications regarding the applicability of neural networks.

Appendix D: SpecNet Model Details

D.1 SpecNet Model Summary

The following is the SpecNet model summary. This deviates slightly from the original implementation in that the input and output have a length of 1000 instead of 640.

Table 2: SpecNet Summary.

Layer	Output Shape	Number of Parameters
Input Layer	(1000, 1)	0
Batch Normalization	(1000, 1)	4
ReLu Activation	(1000, 1)	0
1D Convolution Layer 1	(969, 128)	4,224
1D Convolution Layer 2	(954, 64)	131,136
1D Convolution Layer 3	(947, 16)	8,208
1D Convolution Layer 4	(940, 16)	2,064
1D Convolution Layer 5	(933, 16)	2,064
Dense Layer 1	(933, 32)	544
Dense Layer 2	(933, 16)	528
Flatten	(14928)	0
Dropout Layer (0.25)	(14928)	0
Dense Layer 3	(1000)	14,929,000
Output Layer	(1000, 1)	0

Note: the batch size is implicitly prepended to the output shape of each layer.

D.2 Modified SpecNet Code

```
inputs = tf.keras.Input(shape=(1000,), name='Input')
x = tf.keras.layers.Reshape((1000,1), name='Reshape')(inputs)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.ReLU()(x)
x = tf.keras.layers.Conv1D(128, activation = 'relu', kernel_size = (32))(x)
x = tf.keras.layers.Conv1D( 64, activation = 'relu', kernel_size = (16))(x)
x = tf.keras.layers.Conv1D( 16, activation = 'relu', kernel_size = (8))(x)
x = tf.keras.layers.Conv1D( 16, activation = 'relu', kernel_size = (8))(x)
x = tf.keras.layers.Conv1D( 16, activation = 'relu', kernel_size = (8))(x)
x = tf.keras.layers.Dense(32, activation = 'relu',
      kernel_regularizer=tf.keras.regularizers.l1_l2(l1 = 0, l2=0.1))(x)
x = tf.keras.layers.Dense(16, activation = 'relu',
      kernel_regularizer=tf.keras.regularizers.l1_l2(l1 = 0, l2=0.1))(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dropout(.25)(x)
outputs = tf.keras.layers.Dense(1000, activation='relu')(x)
model = tf.keras.Model(inputs=inputs, outputs=outputs, name='SpecNet')
```

D.3 SpecNet Copyright Notice

SpecNet is provided with the following open source MIT License which allows anyone to use, modify, and publish substantial portions of the code so long as the copyright notice is included.

Appendix E: VECTOR Model Details

E.1 VECTOR Model Summaries

The following two tables demonstrate the summaries for the VECTOR-8 and VECTOR-16 models. The "Operation" column refers to the convolution that is applied to the that layer, where K is the kernel size, C_{in} is the number of input channels, C_{out} is the number of output channels, and S is the stride length.

Table 3: VECTOR-8 Summary

Stage		Operation (K, C_{in}, C_{out}, S)	Output Shape	Number of Parameters
Input		—	(1000, 1)	0
Encoder	Layer 1	(8, 1, 64, 1)	(993, 64)	832
	Layer 2	(8, 64, 128, 2)	(493, 128)	66,176
	Layer 3	(8, 128, 256, 2)	(243, 256)	263,424
	Layer 4	(8, 256, 512, 2)	(118, 512)	1,051,136
Latent Space		—	(118, 512)	0
Decoder	Layer 5	(8, 512, 256, 2)	(243, 256)	1,049,856
	Layer 6	(8, 256, 128, 2)	(493, 128)	262,784
	Layer 7	(8, 128, 64, 2)	(993, 64)	65,856
	Layer 8	(8, 64, 1, 1)	(1000, 1)	513
Output		—	(1000,1)	0

Note: the batch size is implicitly prepended to the output shape of each layer.

Table 4: VECTOR-16 Summary.

Stage		Operation (K, C_{in}, C_{out}, S)	Output Shape	Number of Parameters
Input		—	(1000,1)	0
Encoder	Layer 1	(8, 1, 64, 1)	(993, 64)	832
	Layer 2	(8, 64, 128, 2)	(493, 128)	66176
	Layer 3	(8, 128, 256, 2)	(243, 256)	263424
	Layer 4	(8, 256, 512, 2)	(118, 512)	1051136
	Layer 5	(8, 512, 1024, 2)	(56, 1024)	4199424
	Layer 6	(8, 1024, 2048, 2)	(25, 2048)	16787456
	Layer 7	(8, 2048, 2048, 1)	(18, 2048)	33564672
	Layer 8	(8, 2048, 2048, 1)	(11, 2048)	33564672
Latent Space		—	(11, 2048)	0
Decoder	Layer 9	(8, 2048, 2048, 1)	(18, 2048)	33564672
	Layer 10	(8, 2048, 2048, 1)	(25, 2048)	33564672
	Layer 11	(8, 2048, 1024, 2)	(56, 1024)	16782336
	Layer 12	(8, 1024, 512, 2)	(118, 512)	4196864
	Layer 13	(8, 512, 256, 2)	(243, 256)	1049856
	Layer 14	(8, 256, 128, 2)	(493, 128)	262784
	Layer 15	(8, 128, 64, 2)	(993, 64)	65856
	Layer 16	(8, 64, 1, 1)	(1000, 1)	513
Output		—	(1000, 1)	0

Note: the batch size is implicitly prepended to the output shape of each layer.

E.2 Customized VECTOR Code

The original VECTOR models were implemented using PyTorch, a machine learning framework for Python. In this work the code is repurposed for use with the TensorFlow framework. This does not fundamentally change the operation of the models since both

frameworks operate using the same underlying technology. To use the code below, a working version of TensorFlow must be installed.

The following is the code defining the encoder and decoder.

```
class EncoderLayer(tf.keras.layers.Layer):
    def __init__(self, kernel_size, input_channels, output_channels, stride,
                 name=None):
        super().__init__(name=name)
        self.conv = tf.keras.layers.Conv1D(
            filters=output_channels,
            kernel_size=kernel_size,
            strides=stride
        )
        self.relu = tf.keras.layers.ReLU()
        self.norm = tf.keras.layers.BatchNormalization()

    def call(self, x):
        x = self.conv(x)
        x = self.relu(x)
        x = self.norm(x)
        return x


class DecoderLayer(tf.keras.layers.Layer):
    def __init__(self, kernel_size, input_channels, output_channels, stride,
                 output_padding=1, name=None):
        super().__init__(name=name)
        self.conv = tf.keras.layers.Conv1DTranspose(
            filters=output_channels,
            kernel_size=kernel_size,
            strides=stride,
            output_padding=output_padding # Add output padding to decoder
        )
        self.norm = tf.keras.layers.BatchNormalization()
        self.relu = tf.keras.layers.ReLU()

    def call(self, x):
        x = self.conv(x)
        x = self.norm(x)
        x = self.relu(x)
        return x
```

VECTOR-8 Code

```
skip_connections = False
layer1 = EncoderLayer(8,   1,  64, 1, name="Layer_1")
layer2 = EncoderLayer(8,  64, 128, 2, name="Layer_2")
layer3 = EncoderLayer(8, 128, 256, 2, name="Layer_3")
layer4 = EncoderLayer(8, 256, 512, 2, name="Layer_4")
layer5 = DecoderLayer(8, 512, 256, 2, name="Layer_5")
layer6 = DecoderLayer(8, 256, 128, 2, name="Layer_6")
layer7 = DecoderLayer(8, 128,  64, 2, name="Layer_7")
layer8 = tf.keras.layers.Conv1DTranspose(filters=1, kernel_size=8, strides=1,
                activation='sigmoid', name="Layer_8")
add = tf.keras.layers.Add()

inputs = tf.keras.Input(shape=(1000,), name='Input')
reshaped_inputs = tf.keras.layers.Reshape((1000,1,), name='Reshape')(inputs)
e1 = layer1(reshaped_inputs)
e2 = layer2(e1)
e3 = layer3(e2)
e4 = layer4(e3)
# Latent space
d1 = layer5(e4)
d2 = layer6(add([d1,e3]) if skip_connections else d1)
d3 = layer7(add([d2,e2]) if skip_connections else d2)
d4 = layer8(add([d3,e1]) if skip_connections else d3)
outputs = tf.keras.layers.Flatten(name="Output")(d4)
model = tf.keras.Model(inputs=inputs, outputs=outputs, name='VECTOR-8')
```

VECTOR-16 Code

```
skip_connections = False
layer1  = EncoderLayer(8,    1,   64, 1, name="Layer_1")
layer2  = EncoderLayer(8,   64,  128, 2, name="Layer_2")
layer3  = EncoderLayer(8,  128,  256, 2, name="Layer_3")
layer4  = EncoderLayer(8,  256,  512, 2, name="Layer_4")
layer5  = EncoderLayer(8,  512, 1024, 2, name="Layer_5")
layer6  = EncoderLayer(8, 1024, 2048, 2, name="Layer_6")
layer7  = EncoderLayer(8, 2048, 2048, 1, name="Layer_7")
layer8  = EncoderLayer(8, 2048, 2048, 1, name="Layer_8")
layer9  = DecoderLayer(8, 2048, 2048, 1, name="Layer_9",  output_padding=0)
layer10 = DecoderLayer(8, 2048, 2048, 1, name="Layer_10", output_padding=0)
layer11 = DecoderLayer(8, 2048, 1024, 2, name="Layer_11", output_padding=0)
layer12 = DecoderLayer(8, 1024,  512, 2, name="Layer_12", output_padding=0)
layer13 = DecoderLayer(8,  512,  256, 2, name="Layer_13")
layer14 = DecoderLayer(8,  256,  128, 2, name="Layer_14")
layer15 = DecoderLayer(8,  128,   64, 2, name="Layer_15")
layer16 = tf.keras.layers.Conv1DTranspose(filters=1, kernel_size=8, strides=1,
                  activation='sigmoid', name="Layer_16")
add = tf.keras.layers.Add()

inputs = tf.keras.Input(shape=(1000,), name='Input')
reshaped_inputs = tf.keras.layers.Reshape((1000,1,), name='Reshape')(inputs)
e1 = layer1(reshaped_inputs)
e2 = layer2(e1)
e3 = layer3(e2)
e4 = layer4(e3)
e5 = layer5(e4)
e6 = layer6(e5)
e7 = layer7(e6)
e8 = layer8(e7)
# Latent space
d1 = layer9(e8)
d2 = layer10(add([d1,e7]) if skip_connections else d1)
d3 = layer11(add([d2,e6]) if skip_connections else d2)
d4 = layer12(add([d3,e5]) if skip_connections else d3)
d5 = layer13(add([d4,e4]) if skip_connections else d4)
d6 = layer14(add([d5,e3]) if skip_connections else d5)
d7 = layer15(add([d6,e2]) if skip_connections else d6)
d8 = layer16(add([d7,e1]) if skip_connections else d7)
outputs = tf.keras.layers.Flatten(name="Output")(d8)
model = tf.keras.Model(inputs=inputs, outputs=outputs, name='VECTOR-16')
```

E.3 VECTOR Copyright Notice

VECTOR is provided with the following open source MIT License which allows anyone to use, modify, and publish substantial portions of the code so long as the copyright notice is included.

Appendix F: Training Code

The following code was used to train each of the models.

```
model.compile(loss='mse', optimizer='Adam', metrics=['mean_absolute_error'])

train_size = 25600
batch_size = 256
epochs = 10

train_sequencer = DataSequencer(batch_size, train_size)

model_history = model.fit(
    x=train_sequencer,
    batch_size=batch_size,
    epochs=epochs,
    verbose=2
)
```