Reevaluating Convolutional Neural Networks for Spectral Analysis: A Focus on Raman Spectroscopy

Deniz Soysal,[†] Xabier García–Andrade,[†] Laura E. Rodriguez,[‡] Pablo Sobron,[¶] Laura M. Barge,[§] and Renaud Detry[*,‖,⊥]

†*KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium*
‡*Lunar & Planetary Institute, Universities Space Research Association, 3600 Bay Area Boulevard, Houston, TX 77058, USA*
¶*Impossible Sensing, 20 South Sarah Street, St. Louis, MO 63108, USA*
§*Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, La Cañada Flintridge, CA 91011, USA*
‖*KU Leuven, Dept. Electrical Engineering, Research Unit Processing Speech and Images (PSI), Kasteelpark Arenberg 10, 3001 Leuven, Belgium*
⊥*KU Leuven, Dept. Mechanical Engineering, Research Unit Robotics, Automation and Mechatronics (RAM), Celestijnenlaan 300, 3001 Leuven, Belgium*

E-mail: renaud.detry@kuleuven.be

Abstract

Autonomous Raman instruments deployed on Mars rovers, deep-sea landers, and mobile field robots must interpret *raw* spectra that are distorted by fluorescence baselines, peak shifts, and limited ground-truth labels. Using rigorously documented subsets of the RRUFF mineral database, we systematically evaluate one-dimensional convolutional neural networks (CNNs) and report four practical advances:

(i) Reproducible, baseline-independent classification: Compact end-to-end CNNs surpass k-nearest-neighbors and support-vector classifiers built on handcrafted peak features, eliminating background-correction and peak-picking stages. Unlike earlier work, we isolate the contribution of learned features and ensure full reproducibility by releasing all data splits and preprocessing scripts.

(ii) Pooling controlled robustness: By adjusting a single pooling parameter, CNNs accommodate Raman shift displacements up to $30\,\mathrm{cm}^{-1}$, enabling a practical trade-off between translational invariance and class resolution, aligned with instrument stability and spectral variability.

(iii) Label-efficient learning: Semi-supervised generative adversarial networks and contrastive pretraining improve classification accuracy by up to 11 % when only 10 % of labels are available. While smaller than gains in vision tasks—due to Raman spectra's lower complexity—these methods remain valuable for autonomous deployments with limited annotation.

(iv) Constant time adaptation: Freezing the CNNs backbone and retraining only the softmax layer transfers the model to unseen minerals at $\mathcal{O}(1)$ inference cost, outperforming Siamese networks on resource-limited robotic processors.

The resulting workflow—*train directly on raw spectra, tune pooling to instrument tolerance, add semi-supervision when labels are scarce, and fine-tune lightly for new targets*—offers a

practical path toward robust, low-footprint Raman classification in autonomous field settings.

Keywords

Deep learning, science autonomy, minerals, semi-supervised learning, ocean worlds, astrobiology, deep-sea exploration

1 Introduction

Raman spectroscopy provides fast, nondestructive, *in situ* chemical fingerprints that enable identification of minerals, organics, and biomaterials. A Raman spectrum is produced by measuring the inelastic scattering of light, where the scattered photons correspond to molecular vibrations unique to the material's chemical structure, producing diagnostic peaks but often with weak signals and fluorescence backgrounds that complicate analysis. In the last decade, machine learning (ML) paired with spectroscopy—including Raman—has accelerated discoveries across analytical chemistry, biology, and planetary science by mining large, nonlinear spectral datasets (1–7). This progress reflects better computation, accessible ML libraries, and instruments that generate high-dimensional outputs (Raman, NMR, LC–MS).

Machine learning methods Spectral ML generally falls into supervised and unsupervised paradigms. Supervised learning relies on labeled datasets to train models for classification or regression tasks, proving essential for applications requiring precise predictions such as early disease diagnosis (8), forensic residue analysis (9), and microplastics monitoring (10). On the other hand, unsupervised learning exploits unlabeled data to reveal latent structure.

ML for autonomous exploration In Earth and planetary missions, supervised ML underpins scientific autonomy on robots operating from deep-sea vents to extraterrestrial terrains, where spectrometers guide real-time decisions on sampling and goal replanning (11–17). Limited storage and bandwidth force onboard prioritization of what to transmit or retain (18, 19). Emphasis on autonomy spans deep-sea mining (20), Mars missions (19, 21), and planned explorations of icy ocean worlds (18, 22).

Challenges in applying machine learning to spectral data Traditional ML techniques, such as k-nearest neighbors (KNN) and support vector machines (SVM), typically rely on heavy preprocessing to manage high dimensionality, fluorescence baselines, and noise (1, 7, 23, 24). Baseline correction is critical for Raman spectroscopy (23), with analogous handling for the Bremsstrahlung effect in LIBS (25) and chromatographic drift (26). Yet expert-driven preprocessing can inject bias, and heterogeneous pipelines hinder reproducibility. A second constraint is label scarcity: obtaining mineral ground truth (XRD/chemistry) is costly and labor-intensive. Ground-truthing measurements from surface-based techniques such as Raman, LIBS, and Vis-IR spectroscopy further complicate the process, as they often require detailed comparisons between bulk and surface analyses. Biomedical and field datasets are further limited by permits, logistics, and budgets. These realities motivate label-efficient models and reproducible workflows that leverage unlabeled data. Additionally, approaches tailored to the unique characteristics of spectral data could help overcome these limitations, enabling more efficient and accurate use of both labeled and unlabeled datasets.

Deep learning methods Deep learning, a branch of ML, utilizes a series of nonlinear transformations, often implemented as neural networks, to extract meaningful insights from data. Deep learning models learn nonlinear representations directly from data and can operate on *raw* spectra, reducing reliance on handcrafted preprocessing (23). Neural networks encompass diverse architectures but share a common structure comprising three key components: an input layer for data ingestion, one or more hidden layers where nonlinear transformations are applied, and an output layer that gen-

erates predictions. Fully connected deep neural networks (DNNs) interconnect every node between layers, but they scale poorly as input dimensionality grows. Convolutional neural networks (CNNs), by contrast, use local kernels and pooling to extract compact, hierarchical features and generalize better on images and spectral signals (1, 4, 7, 23, 27–31). CNNs' built-in inductive biases (such as locality and translational invariance) align particularly well with the characteristics of image data and facilitate efficient feature extraction.

Challenges for deep learning in spectral data Unlike images, small Raman peak shifts can be chemically meaningful; indiscriminate translational invariance is harmful (32). CNNs' invariance must therefore be controlled via architectural adjustments and hyperparameter tuning to be robust to instrument drift yet sensitive to class-defining displacements. Although CNNs need fewer labels than fully connected DNNs, they remain more data-hungry than KNN or SVM because they *learn* features rather than rely on engineered ones. Label-efficient strategies are thus important: domain-aware augmentation (23, 33, 34), semi-/self-supervised learning to exploit unlabeled spectra (35, 36), and transfer learning to adapt pretrained backbones to new datasets (32, 37). Although these strategies show promise, their effectiveness in the context of Raman spectroscopy and other spectral data remains an area requiring further investigation.

Applications of ML in Raman spectroscopy Traditional Raman pipelines have used multivariate discriminant analysis (38), partial least-squares regression (39), KNN (40), and SVM (41). More recently, CNNs have emerged as a more effective approach for analyzing Raman spectra (23, 42). To address the challenge of limited labeled data, Liu *et al.* (24) proposed a representation learning approach using Siamese networks (43), shifting from multi-class classification to binary classification. Despite progress, two gaps remain: (i) image-centric CNNs inductive biases are not fully aligned with Raman peak physics, and (ii) most methods do not fully leverage the abundant unlabeled spectra. Addressing both is essential especially in label-scarce settings.

Scope of This Study We build upon prior work by focussing on understanding the role of inductive biases in CNNs' architectures and exploring data-efficient strategies such as semi-supervised learning and transfer learning. Unlike previous studies that primarily benchmark models, our work provides an in-depth analysis of CNNs robustness to spectral shifts, investigates the complexity of low-level spectral features, and offers practical guidelines for deploying CNNs in data-scarce environments.

Using rigorously documented subsets of the RRUFF database (44), our objectives are fourfold:

1. **Investigating CNNs Inductive Biases:** Analyze how the inherent inductive biases of CNNs' architectures impact Raman spectral classification, and propose architectural adjustments to enhance performance.

2. **Benchmarking Against Traditional Methods:** Compare CNNs with classical machine learning approaches, including SVM and KNN, on both raw and processed Raman spectra.

3. **Exploring semi-supervised learning:** Evaluate semi-supervised methods such as semi-supervised generative adversarial networks (SGANs) and contrastive learning to improve accuracy by leveraging unlabeled spectral data.

4. **Analyzing low-level spectral features:** Examine the complexity and structure of low-level Raman features to understand their role in model performance and guide architecture design.

To keep the narrative clear, we interleave methodological discussion with results. Section 2 introduces our RRUFF subsets; Section 3 establishes SVM/KNN baselines; Section 4 shows CNNs' superiority on raw spectra; Section 5 tunes inductive biases for shift

robustness; Section 6 develops semi-supervised approaches; Section 7 investigates feature complexity; and Section 8 demonstrates lightweight transfer learning for new mineral classes.

2 The Database

Our experiments utilize the RRUFF Database[1], the largest publicly available mineral spectral database (44). The database currently holds spectra for 4 216 curated specimens (covering 2 351 mineral species out of 6 006 IMA-approved minerals).

RRUFF content and provenance The RRUFF Project was created to provide "a complete reference set of Raman spectra, X-ray diffraction patterns and electron-microprobe analyses for every mineral species" (44). Each database record corresponds to a single, archived specimen with a unique RRUFF ID and includes (i) Raman spectra acquired at 532 nm (often additional wavelengths) in both *Raw* and baseline-corrected *Processed* forms; (ii) a powder-XRD pattern used to confirm phase purity; (iii) full chemical analysis and ideal formula; (iv) detailed acquisition metadata (laser power, spot size, spectrometer, crystal orientation, temperature); and (v) locality information and a photomicrograph. Because every spectrum is traceable to a specimen whose identity has been independently validated by XRD and chemistry, the mineral name supplied by RRUFF constitutes a ground-truth label for supervised learning.

What a spectrum represents A Raman entry in RRUFF is a one-dimensional vector $I(\tilde{\nu})$ of photon counts (or arbitrary intensity units) as a function of Raman shift $\tilde{\nu}$ (cm^{-1}) measured from a single laser spot on the specimen. Most specimens are analyzed in multiple modes: high-resolution scans (narrow range, $\sim$ 70–1 400 cm^{-1}) that resolve diagnostic peaks, low-resolution scans (broad range, 70–6 500 cm^{-1}) that capture lattice modes and overtones, and optionally oriented measurements with the crystallographic axes aligned to the laser polarization.

[1] `https://rruff.info/`

Terminology used in the remainder of this paper

- **Spectrum**: a single Raman vector $I(\tilde{\nu})$ as defined above.
- **Labeled spectrum**: a spectrum whose mineral species (e.g. "quartz") is provided to the model during training or evaluation; in practice this means the species tag supplied by RRUFF.
- **Unlabeled spectrum**: a spectrum for which the species tag is withheld, either deliberately (e.g. to create semi-supervised splits) or because ground-truth is absent in a field-deployment scenario.

We will consistently use the terms labeled and unlabeled in this sense throughout the paper. For every Raman measurement, RRUFF stores a text file whose file name indicate the processing state—ending in "..._Raman_Data_**RAW**.txt" or "..._Raman_Data_**Processed**.txt". Throughout this paper we refer to these two sets simply as

- **RRUFF-raw**: spectra as recorded in the RAW format, without baseline preprocessing;
- **RRUFF-clean**: baseline-corrected spectra (processed format, where baselines have been removed by CrystalSleuth(44)).

2.1 Subset of the RRUFF database used in this study

We consider two subsets of the RRUFF database:

- **Low-resolution unoriented subset:** Downloaded from `https://rruff.info/zipped_data_files/`

raman/LR-Raman.zip, containing unoriented low-resolution samples. Classes with fewer than eight samples are pruned, and only the RRUFF-raw spectra are retained.

- **High-resolution oriented subset:** Downloaded from https://rruff.info/zipped_data_files/raman/excellent_oriented.zip, containing oriented high-resolution samples. Same pruning is applied, and only the RRUFF-clean spectra are retained.

This results in two datasets: one composed of RRUFF-raw spectra and another of RRUFF-clean spectra. These datasets are then split into training and test sets. The specific subsets used in our experiments can be downloaded from https://github.com/denizsoysal/Raman_spectra_data.

2.2 Preprocessing

In addition to the baseline correction performed in the RRUFF-clean spectra by the database maintainers, we apply our own preprocessing pipeline systematically to both RRUFF-raw and RRUFF-clean data. These steps are fully automated and applied in batch mode across the entire dataset, without any human intervention at the sample level.

Class Pruning Classes with fewer than $n_{\min} = 8$ samples are removed to maintain sufficient training data per category.

Wavelength Range Selection Raman shifts outside the $(200, 1600)\,\mathrm{cm}^{-1}$ range are discarded. This decision is motivated by the observation that peaks are concentrated in the lower Raman shift values, whereas higher shifts provide limited discriminative information. Additionally, the lowest Raman shift region corresponds to intermolecular vibrations, which are not discriminative for mineral classification. Based on this observation, the models are restricted to the $\nu \in (200, 1600)\,\mathrm{cm}^{-1}$ domain.

Interpolation and Resampling To standardize input dimensions and spectral resolution, each raw spectrum's shift–intensity pairs are projected onto a common Raman-shift grid spanning 200-1600 cm^{-1} in 1 cm^{-1} steps. These bounds and step size are chosen based on statistical summaries of all spectra, including minimum/maximum shift ranges and the average sampling resolution. This yields 1401 points, which we then truncate to 1392 points (the nearest lower multiple of 16) to satisfy GPU alignment constraints. Projection is performed via linear interpolation, with any values outside a spectrum's original range set to zero. No additional smoothing or filtering is applied, preserving spectral fidelity. The resulting 1392-dimensional intensity vector ensures uniform resolution, consistent dimensionality, and efficient batching for neural network training.

Normalization All spectra are normalized to ensure consistent intensity scaling, reducing variability introduced by acquisition conditions.

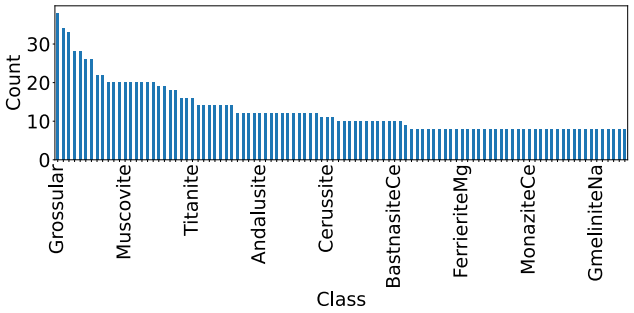
We emphasize that the preprocessing steps described above are computationally inexpensive and applied in batch mode across the entire dataset. No human intervention is required on a per-sample basis, which ensures reproducibility and mitigates the risks of manual bias typically associated with expert-guided preprocessing.

2.3 Final Dataset

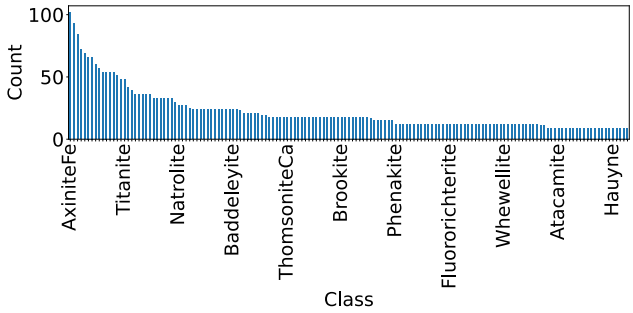
After pruning classes with fewer than eight spectra and applying the pre-processing pipeline, the corpus splits into two subsets:

- **RRUFF-clean**: 3 425 spectra, 155 mineral classes (mean 22.1 spectra per class).
- **RRUFF-raw**: 1 311 spectra, 102 mineral classes (mean 12.9 spectra per class).

Both subsets are highly imbalanced (Figure 1), with most species represented by fewer than 20 spectra. To mitigate this, we apply class weights in the cross-entropy loss, forcing the network to pay proportionally more attention to under-represented classes.



(a) Class-size histogram for the RRUFF-raw subset.



(b) Class-size histogram for the RRUFF-clean subset.

Figure 1: Number of spectra per mineral class after all pre-processing steps. The imbalanced distribution motivates the use of class-weighted loss in model training.

2.4 Illustrative spectral variability

Figure 2 plots dolomite with four other carbonate minerals whose Raman signatures are almost super-imposable. All five spectra feature the characteristic carbonate band at $\sim 1085\,\mathrm{cm}^{-1}$; the principal peaks differ by less than $10\,\mathrm{cm}^{-1}$, creating a fine-resolution challenge for any classifier. Figure 3 instead compares dolomite with four minerals from chemically different groups (arsenates, hydroxides, silicates). Here, the spectra share no dominant bands, illustrating markedly different vibrational chemistries. Taken together, the two figures span the extremes of our database, from near-duplicate spectra to entirely distinct signatures.

2.5 Data augmentation strategy

Given the characteristics of Raman spectra, augmentation techniques must be carefully designed to preserve spectral peaks. We employ transformations that retain the integrity of Raman features, as detailed in Section 6.1. Data augmentation is applied in Section 4.2 to evaluate the performance of CNNs on augmented data and in Section 6.1 in the context of semi-supervised learning.

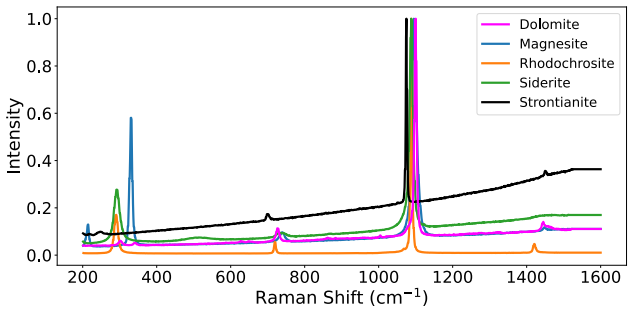
2.6 Implementation and Training Environment

All deep-learning models were implemented in PyTorch. Classical machine-learning baselines (SVM and KNN) were trained with scikit-learn. Training was performed on a laptop equipped with an NVIDIA RTX 3070 GPU. Typical training time per data split was $\sim$1–2 hours for deep-learning experiments.

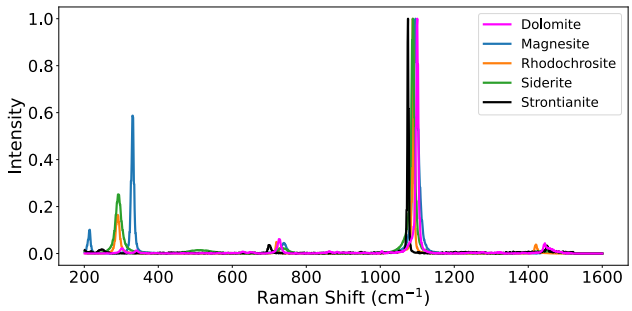
3 Classification using SVM and KNN

Before introducing advanced neural network-based approaches, it is essential to establish a performance baseline using traditional machine learning methods for Raman spectral classification. Support vector machines (SVM) and k-nearest neighbors (KNN) are two widely used techniques. SVM operates by finding an optimal hyperplane that separates data points into distinct classes while maximizing the margin between them. KNN, in contrast, is a distance-based approach that assigns an unknown data point x to the most common class among its k nearest neighbors in the dataset X. (Refer to Supporting Information for further details on SVM and KNN.)

In this section, we implement both SVM and KNN classifiers not only to explore their performance but, more importantly, to serve as explicit baselines for our subsequent experiments



(a) RRUFF-raw spectra, showing fluorescence baseline.



(b) RRUFF-clean spectra, with baseline removed.

Figure 2: RRUFF-raw and RRUFF-clean spectra for dolomite ($CaMg(CO_3)_2$) plotted alongside four compositionally related carbonate minerals—magnesite ($MgCO_3$), rhodochrosite ($MnCO_3$), siderite ($FeCO_3$), and strontianite ($SrCO_3$). The shared carbonate ν_1 symmetric stretch at 1080–1100 cm^{-1} and lattice modes near 300 cm^{-1} illustrate the high intra-group spectral similarity, while the RRUFF-raw scans highlight fluorescence backgrounds removed in the RRUFF-clean scans.

with deep learning models (see Section 4). Establishing these baselines allows us to rigorously quantify the improvements introduced by convolutional neural networks and to contextualize our findings within existing machine learning frameworks for spectral data.

Prior work by Widjaja *et al.* (41) demonstrated the application of SVM for Raman spectral classification, employing preprocessing techniques such as dimensionality reduction to mitigate noise and baseline effects from fluorescence. In this study, we implement both SVM and KNN classifiers and evaluate their performance using feature extraction techniques tailored for Raman spectra. Our findings indicate that while SVM and KNN achieve satisfactory results on RRUFF-clean data, their performance degrades significantly when applied to RRUFF-raw spectral data.

3.1 Methodology

Feature extraction plays a crucial role in determining the effectiveness of machine learning classifiers. To ensure high classification performance, the extracted features must be highly informative regarding class distributions.

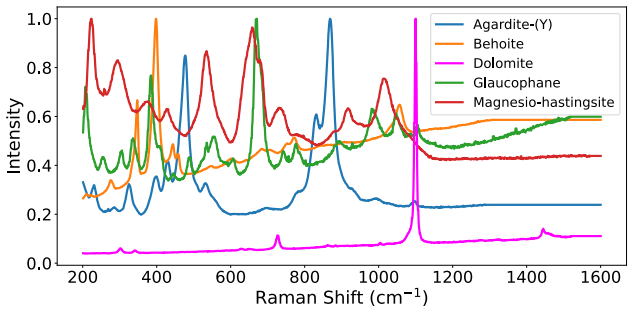
As detailed in the Supporting Information, the primary features relevant to mineral classification in Raman spectroscopy are the *positions of spectral peaks*. To extract these peaks, we evaluated two peak detection methods:

1. **Wavelet-based peak detection:** Uses a continuous wavelet transform to identify spectral peaks by matching the signal shape with a predefined wavelet function (`signal.find_peaks_cwt` from SciPy (45)).

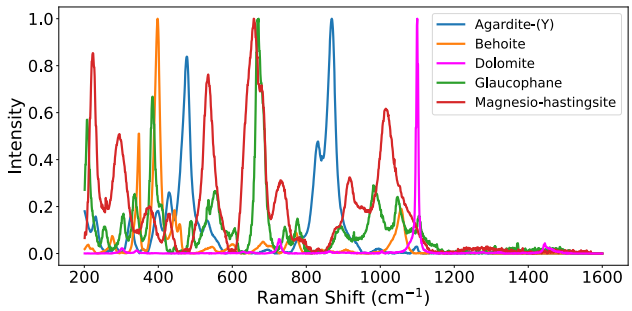
2. **Local maxima detection:** Identifies peaks by comparing each value with its neighboring points (`signal.find_peaks` from SciPy).

We performed a grid search to determine the optimal parameters for each method. Table 1 presents the classification accuracy of KNN using each peak detection approach. The wavelet-based method consistently outperformed the local maxima approach, leading us to select it as our primary peak detection technique.

Feature vector construction The wavelet detector returns a set of Raman–shift positions at which peaks occur. Because each spectrum contains a different number of peaks, we convert that variable-length list into a fixed-length feature vector by binning the shift axis and counting how many peaks fall inside each bin. We adopt a bin width of 12 cm^{-1}, guided by the typical variability of the carbonate ν_1 symmetric-stretch band (centered near



(a) RRUFF-raw spectra, showing fluorescence baseline.



(b) RRUFF-clean spectra, with baseline removed.

Figure 3: RRUFF-raw and RRUFF-clean spectra for dolomite ($CaMg(CO_3)_2$) plotted alongside four spectrally dissimilar minerals: agardite-(Y) ($YCu_6(AsO_4)_3(OH)_6 \cdot 3\,H_2O$), behoite ($Be(OH)_2$), glaucophane ($Na_2(Mg_3Al_2)Si_8O_{22}(OH)_2$), and magnesio-hastingsite ($NaCa_2(Mg_4Fe)(Si_6Al_2)O_{22}(OH)_2$). The dolomite trace shows the characteristic carbonate ν_1 band at 1080–1100 cm^{-1}, whereas the comparison minerals exhibit dominant arsenate or silicate stretches and entirely different lattice modes. Together with Figure 2, this panel demonstrates the spectral variability in the dataset, from very closely related classes that differ by subtle peak shifts to minerals that are spectrally distant.

Table 1: Comparison of peak detection methods for KNN classification. Wavelet-based detection, using shape-matching with Ricker wavelets, achieves higher accuracy than local maxima, demonstrating its robustness to spectral noise and baseline variations.

Method	Wavelet-Based	Local Maxima-Based
Function	`find_peaks_cwt`	`find_peaks`
Peak detection strategy	Shape-matching using Ricker wavelets	Intensity-based local maxima
Best Parameters	Width: [10, 20] Wavelet: Ricker	Width: 10 Prominence: max(intensity)/len(signal)
Top-1 Accuracy (KNN)	70%	59%

1085 cm^{-1}), which shifts by up to $\sim$ 10 cm^{-1} between closely related species. This choice is therefore narrow enough to distinguish those fine spectral differences yet broad enough to keep the feature space compact and avoid the curse of dimensionality associated with very high-dimensional feature spaces. Given the input vector of 1392 dimensions, partitioning it into 12 cm^{-1} intervals produces exactly 116 bins; each spectrum is therefore encoded as a 116-dimensional peak-count vector, independent of the number of detected peaks. Figure 4 illustrates the procedure; the resulting vectors serve as inputs to the KNN and SVM classifiers.

3.2 Experiments

To determine optimal hyperparameters, we performed a grid search for both KNN and SVM. The best-performing configurations were:

- **KNN**: $k = 1$, Euclidean distance[2]
- **SVM**: Regularization parameter $C = 10$, kernel: Radial Basis Function (RBF), kernel coefficient $\gamma = 0.01$.

[2]Using Euclidean distance in the 116-dimensional peak-count space, the mean intra-/inter-class distances are 2.2 / 6.1 for RRUFF-clean and 6.5 / 13.1 for RRUFF-raw. As intra-class variation is low relative to inter-class variation, consulting more than one neighbor adds little useful information and can even mix in other classes; hence $k = 1$ is optimal.

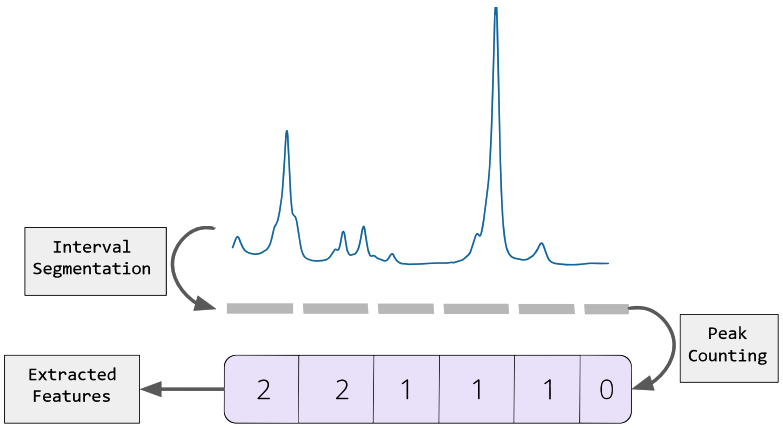


Figure 4: Feature extraction process for traditional ML methods. The Raman spectrum is segmented into intervals and peaks are counted within each interval.

Table 2: Performance of KNN and SVM on RRUFF-raw and RRUFF-clean Raman spectra. Both models show accuracy improvements on RRUFF-clean data.

Dataset	RRUFF-raw	RRUFF-clean
Top-1 Accuracy of SVM (%)	49	72
Top-1 Accuracy of KNN (%)	51	70

Considering the class imbalance in the dataset, we employed stratified five-fold cross-validation and reported the Top-1 accuracy (fraction of spectra whose highest-probability predicted class matches the ground-truth label). Table 2 summarizes the results.

Both models exhibit significantly lower accuracy on RRUFF-raw spectra compared to RRUFF-clean spectra. Our findings highlight the critical role of feature extraction in traditional ML-based Raman spectral classification. When applied to RRUFF-clean spectra, where preprocessing steps have mitigated baseline effects and noise, both KNN and SVM achieve relatively high accuracy (around 70%). However, for RRUFF-raw spectra, these classifiers struggle due to the reliance on explicit peak detection, which becomes unreliable under noise and baseline distortions. Figure 5 illustrates this: the fluorescence baseline hides several diagnostic peaks in the RRUFF-raw spectrum (upper panel) that are recovered in the RRUFF-clean spectrum (lower panel).

An open question remains as to whether the observed performance drop is primarily due to the limitations of feature extraction or the classifiers' decision strategies within the feature space. We address this question in Section 4.2, where we explore whether deep learning models can overcome these challenges by learning more robust features directly from the raw spectral data.

4 Classification using CNNs

As discussed in Section 3, traditional machine learning methods struggle to classify raw Raman spectra effectively. To overcome these limitations, we explore deep learning-based approaches. Specifically, we investigate whether convolutional neural networks (CNNs) can overcome the bottlenecks identified with classical classifiers—namely, the reliance on explicit feature extraction and potential limitations in decision strategies within the feature space. CNNs are a class of deep learning models primarily used for image processing but have also been successfully applied to spectral data. CNNs extract features by convolving a learnable filter (or kernel) with the input signal. Unlike classical convolutional operations, where the filter is predefined, CNNs learn these filters autonomously during training (refer to Supporting Information for more details on CNNs).

Although CNNs are typically designed for 2-

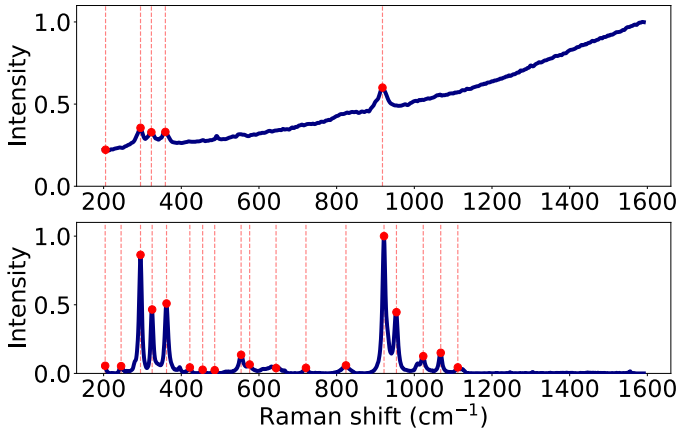


Figure 5: Wavelet peak detection on Andalusite. Upper: RRUFF-raw spectrum—several diagnostic peaks are missed due to fluorescence baseline. Lower: RRUFF-clean spectrum—more peaks are detected, illustrating why peak-based feature extraction works better on the cleaned data.

D image data, in this work, we employ 1-D CNNs, which is well suited for 1-D data, including spectral data. Unlike 2-D CNNs that operate across two spatial dimensions, 1-D CNNs perform convolutions along a single axis.

CNNs, by learning local patterns and compositional hierarchies, offer a natural advantage in processing the structured but noisy signals of Raman spectra. Liu *et al.* (23) demonstrated the effectiveness of deep CNNs for Raman spectral classification, showing that CNNs-based models can learn automatic baseline corrections that are more robust than traditional manual preprocessing. Moreover, the hierarchical nature of CNNs enables them to learn intermediate feature representations, leading to superior classification performance compared to SVMs, particularly for datasets with a large number of classes. The CNNs' architecture used by Liu *et al.*, inspired by the LeNet model (46), is illustrated in Figure 6.

Their study, conducted on the RRUFF database, reported that CNNs outperform conventional classifiers. However, the paper lacks details regarding the specific subsets of the dataset and the preprocessing techniques applied. To provide a reproducible baseline for future research, we replicate their approach while explicitly documenting the dataset partitions and preprocessing steps in Section 2. Our objective is to enhance reproducibility and enable fairer performance comparisons.

4.1 Methodology

We adopt the CNNs' architecture proposed by Liu *et al.*, depicted in Figure 6, with one key modification: the removal of Batch Normalization layers, as preliminary experiments indicated improved performance without them. To assess the effectiveness of CNNs, we compare their performance against a Multilayer Perceptron (MLP) and KNN classifiers. The MLP model shares the same final two layers as the CNNs, but its first three convolutional layers are replaced with fully connected layers; we evaluate three capacities: MLP-small (16, 32, 64 neurons), MLP-mid (32, 64, 128 neurons), and MLP-large (64, 128, 256 neurons).

We evaluate classification performance using the following approaches:

1. CNNs for end-to-end classification.
2. CNNs for feature extraction, followed by KNN for classification.
3. MLP for classification.
4. KNN for classification using handcrafted features.

To transition from approach (1) to approach (2), we remove the final classification layer from the CNNs and use the extracted features as input for a KNN classifier. This setup allows us to further analyze insights from Section 3.2

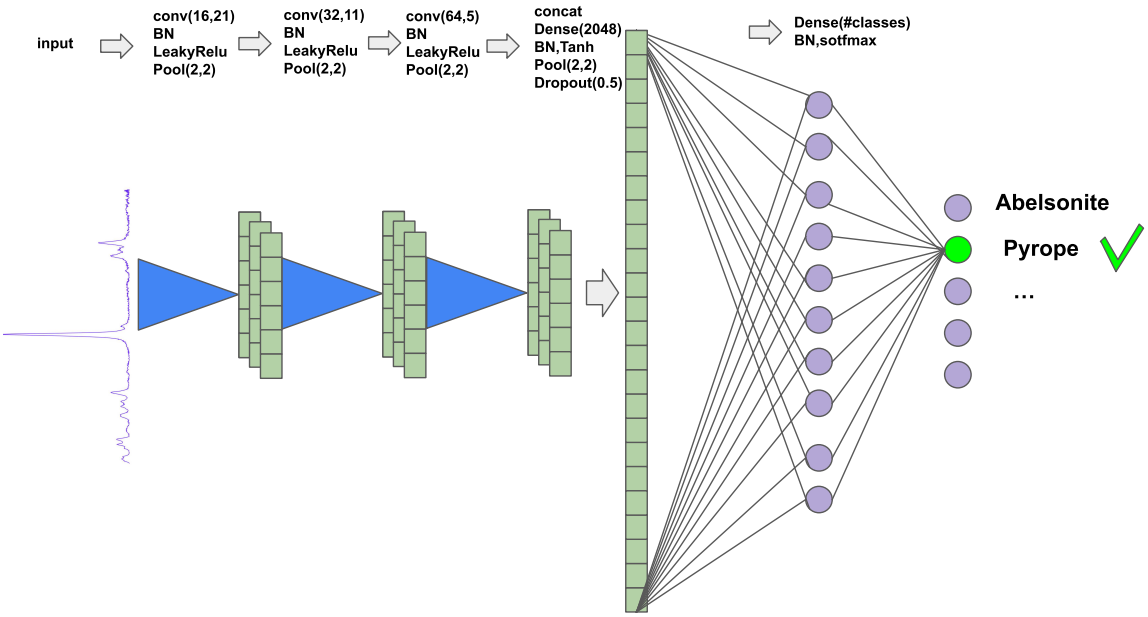


Figure 6: Architecture of the 1-D CNNs model used for Raman spectral classification. The input consists of raw Raman spectra, which pass through three convolutional layers (16, 32, and 64 filters, with kernel sizes of 21, 11, and 5, respectively). Each convolutional layer is followed by Batch Normalization (BN), LeakyReLU activation, and max pooling (2×2). The extracted features are concatenated and processed through a dense layer (2048 neurons) with hyperbolic tangent (Tanh) activation, dropout (0.5), and additional pooling. The final classification layer applies a softmax function to assign the spectrum to one of the mineral classes. This architecture is based on the design proposed by Liu *et al.* (23).

and determine whether KNN's limitations result from its inability to extract meaningful features or from intrinsic weaknesses in the classification algorithm itself.

4.2 Experiments

To evaluate the performance of CNNs for Raman spectral classification, we conducted experiments using a stratified five-fold cross-validation strategy, ensuring balanced class distributions across training and validation sets, as done in Section 3.2.

The CNNs model was trained with an initial learning rate of 0.001, which was dynamically adjusted based on validation performance. If the validation loss did not improve for three consecutive epochs, the learning rate was reduced by a factor of 0.7. Additionally, early stopping was applied to prevent overfitting, terminating training if no improvement was observed for five consecutive epochs.

Figure 7 presents the Top-1 classification accuracy for each method. We observed that:

- CNNs outperform all other approaches, achieving the highest accuracy on both RRUFF-raw and RRUFF-clean datasets.
- CNNs perform better on RRUFF-raw data than on RRUFF-clean data, likely due to their ability to learn robust representations and correct baseline variations automatically (23).
- Data augmentation improves performance, as indicated by increased accuracy across all methods when augmentation was applied (see Section 2.5).

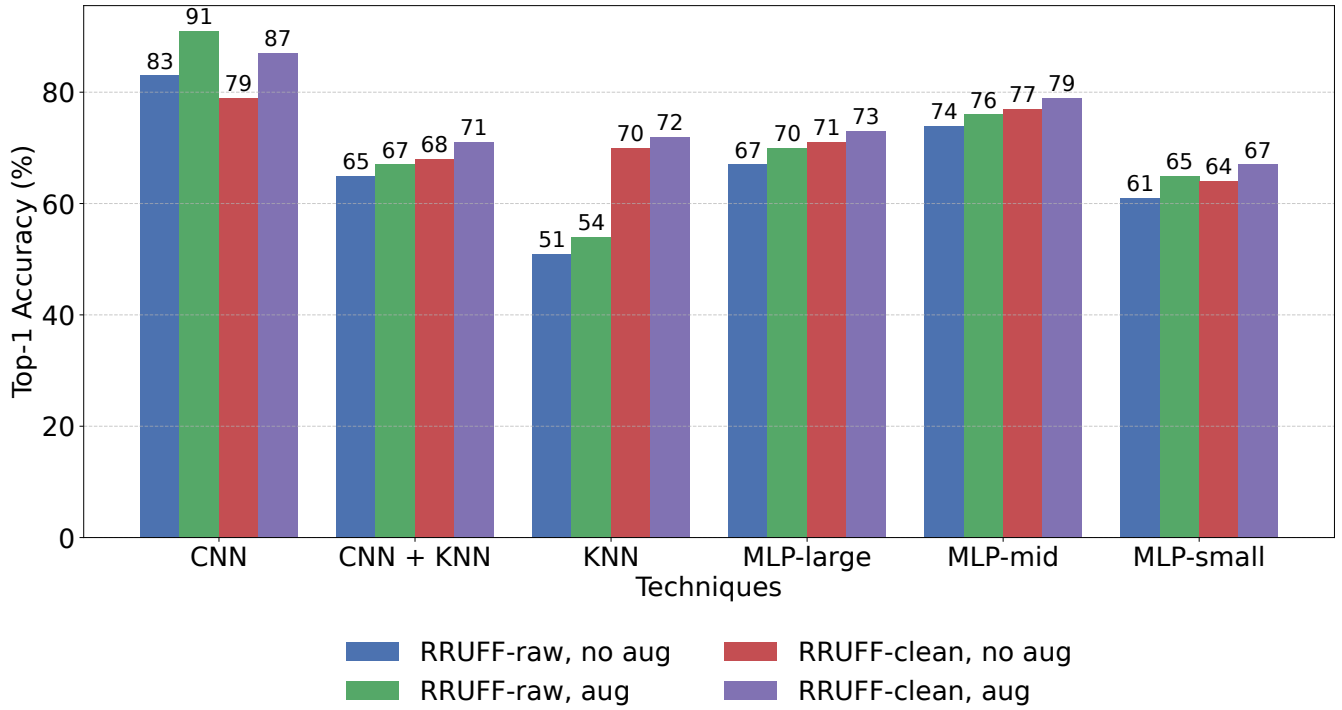


Figure 7: Top-1 accuracy on the test set for CNNs, CNNs+KNN (feature extraction with CNNs followed by KNN classification), MLP, and KNN classifiers. Results are shown for both RRUFF-raw and RRUFF-clean Raman datasets, with and without data augmentation. CNNs achieve the highest performance across all settings, especially with data augmentation.

- CNNs-extracted features improve KNN classification, but CNNs+KNN still underperforms compared to the fully end-to-end CNNs.

This suggests that:

(a) CNNs demonstrate superior feature extraction capabilities compared to traditional approaches such as wavelet-based peak detection and peak counting. This is evident from the improved classification performance when using CNNs as a feature extractor, indicating that CNNs can learn more complex and relevant spectral representations. The ability of CNNs to learn hierarchical features directly from raw spectra also explains the poor performance of KNN and SVM, which rely on manually engineered features.

(b) The lower classification performance of KNN compared to CNNs and MLP suggests that KNN struggles with high-dimensional feature spaces where distance metrics become less discriminative. CNNs and MLPs, on the other hand, leverage learned feature representations to improve class separability. Furthermore, the performance gap between methods (2) and (4) indicates that using CNNs for feature extraction provides a more informative representation than handcrafted features, but KNN remains inherently limited in classification capability.

Learning curves(training and validation loss) We examined the training and validation loss curves (Supporting Information, Figures S9–S16) to assess the learning behavior of the deep learning models. On both datasets, MLP-small shows persistently high training and validation losses with slow decrease (underfitting; Figures S9,S10), whereas MLP-large exhibits a rapid drop in training loss and a widening train–validation gap (overfitting; Figures S13,S14). Across MLP settings, MLP-mid provides the most stable behavior, with con-

vergent losses and a small training/validation loss gap (Figures S11,S12). CNNs achieve the lowest validation loss and fastest convergence on both RRUFF-raw and RRUFF-clean (Figures S15,S16).

Confidence and Calibration For CNNs, we quantified decision certainty via the *confidence gap* (top–1 minus top–2 softmax probability per spectrum). CNNs exhibit well-calibrated margins, with mean confidence gaps of 0.46 on RRUFF-raw and 0.38 on RRUFF-clean. These values reflect the inherent difficulty of the task: the dataset contains spectrally similar minerals, such as those in Figure 2. The model's measured uncertainty on a dataset containing such closely related species is scientifically appropriate, as excessively large confidence would suggest overfitting to training artifacts rather than genuine discriminative learning. The slightly higher gap for raw spectra (0.46 vs. 0.38) parallels their superior classification accuracy, consistent with the idea that retaining full spectral information enables more decisive yet still properly calibrated predictions.

4.2.1 Interpreting CNNs decisions with Grad-CAM

To probe where CNNs find evidence for each mineral, we visualized their internal activation maps with Gradient-weighted Class Activation Mapping (Grad-CAM) (47). For any input spectrum, Grad-CAM back-propagates the prediction score to the last convolutional layer, weights the resulting feature maps by the class-specific gradients, and collapses them to a one-dimensional *importance curve*. We plot that curve as a semi-transparent color bar: warm hues highlight Raman shifts that increase the class score, cool tones those that contribute little.

Across the examples below the warmest regions tend to coincide with well-known diagnostic peaks (e.g. the carbonate ν_1 stretch at $\sim 1085\,\mathrm{cm}^{-1}$) rather than with noise or baseline structure. This pattern indicates that CNNs focus on spectral features that are chemically meaningful. We observe that:

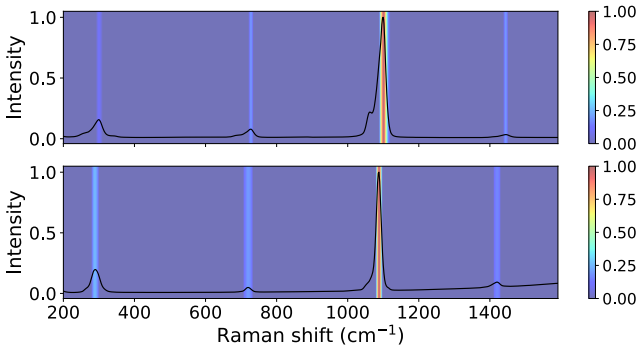


Figure 8: Grad-CAM activation maps for two carbonates. *Top*: Dolomite ($CaMg(CO_3)_2$), correctly classified. *Bottom*: Rhodochrosite ($MnCO_3$), correctly classified.

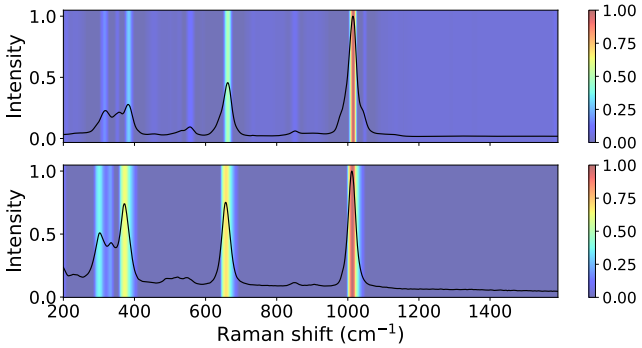


Figure 9: Grad-CAM activation maps for two pyroxenes. *Top*: Diopside ($CaMgSi_2O_6$), misclassified as Hedenbergite. *Bottom*: Hedenbergite ($CaFeSi_2O_6$), correctly classified.

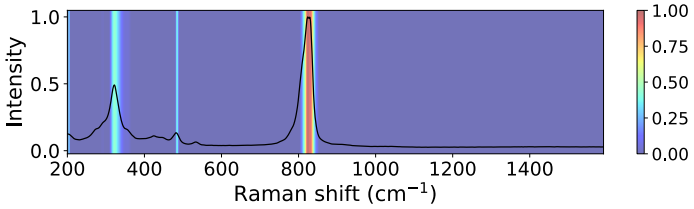


Figure 10: Grad-CAM activation map. Conichalcite ($CaCu(AsO_4)(OH)$), correctly classified.

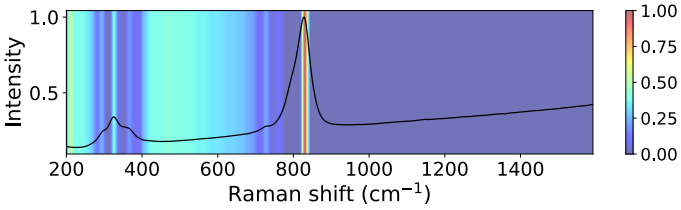


Figure 11: Grad-CAM activation map. Vanadinite ($Pb_5(VO_4)_3Cl$), correctly classified.

The network correctly distinguishes similar carbonates Figure 8 contrasts two carbonate end-members. For both dolomite and rhodochrosite, the hottest bars coincide with the dominant ν_1 C–O stretch (48) at 1097 and 1085 cm^{-1} respectively, plus the weaker lattice ν_4 band near 300 cm^{-1}. Those two regions alone suffice for the model to distinguish Mg from Mn carbonates despite their otherwise similar spectra.

We observe that CNNs do not merely count peaks but encodes where, to the nearest Raman shift, those peaks reside. Thus even a modest 10 cm^{-1} displacement of the strong ν_1 line provides enough evidence for the network to decide between dolomite or rhodochrosite, exactly the fine-grained reasoning that a human analyst would employ.

It is worth noting that the network used here employed a pooling size of 2, which preserves sensitivity to such fine spectral displacements. In Section 5, we explicitly vary the pooling size (2 vs. 64) and show that larger pooling confers robustness to instrumental drift by introducing translational invariance, while smaller pooling retains sensitivity to chemically meaningful shifts. The two results are therefore consistent: depending on the pooling hyperparameter, CNNs can be tuned either for discrimination of closely related spectra or for robustness to noise.

Using peak *absence* as evidence The Grad-CAM maps reveal that CNNs do not merely identify single diagnostic peaks but learn combinations of peaks and their relative intensities, effectively capturing fundamental vibrational coupling patterns rather than memorizing isolated peak positions.

For conichalcite (Figure 10), the network highlights the main AsO_4^{3-} ν_1 stretch at 834 cm^{-1}, alongside smaller lattice modes at 470, 330, and 200 cm^{-1} corresponding to Cu–O and O–As–O vibrations (49).

For vanadinite (Figure 11), the network focuses on the primary VO_4^{3-} ν_1 stretch at 848 cm^{-1} and the peaks around 290–370 cm^{-1}. Interestingly, it also attends to the 400–650 cm^{-1} region—the domain of conichalcite's Cu/As bands—and uses the absence of peaks there to rule out conichalcite in favor of vanadinite.

A failure case: Diopside vs. Hedenbergite Figure 9 shows a typical error when the model is asked to distinguish Mg rich diopside ($CaMgSi_2O_6$) from Fe rich hedenbergite ($CaFeSi_2O_6$). The two Raman spectra display an almost identical three–peak motif at 300–400 cm^{-1} and very similar band shapes elsewhere, so the Grad-CAM heat maps illuminate the same features in both spectra. The subtle shift of the 660 cm^{-1} band is not captured by the model. A larger and more compositionally diverse training set could help CNNs learn to discriminate between these closely related minerals.

5 Understanding CNNs inductive biases for Raman spectra classification

While CNNs show strong empirical performance on Raman spectra, an important question remains: what mechanisms enable these models to effectively process spectroscopic data, which differs fundamentally from image data for which CNNs were originally designed (46)? Convolution operations introduce properties such as translational equivariance, and pooling layers confer a degree of invariance — inductive biases well-suited to images but whose relevance to Raman spectra must be carefully examined.

In this section, we analyze the specific features present in Raman data and explore how the inductive bias of CNNs' architectures can be adapted to better capture the hierarchical and local patterns within spectral signals. This investigation provides deeper insights into the model's decision-making process and guides the design of architectures tailored for spectroscopic analysis.

5.1 Methodology: Linking Raman Characteristics with CNNs Behavior

A key component of the underlying physical mechanism that generates the data must be considered. Raman spectroscopy, as mentioned in the Supporting Information, measures the vibrational modes of a sample. However, the vibrations are susceptible to the experimental conditions. Factors such as temperature, sample impurities or pressure can significantly distort the obtained signal (see Ferraro *et al.* (50) for more details). In addition to alterations in the sample, the experimental equipment can also contribute to spectral distortions. The use of different lasers for the excitation of the sample and spectrometers for processing the signal can also serve as an additional noise source. These distortions in the data manifest primarily as scaling, baseline variation and peak shifts.

- **Scaling:** CNNs inherently handle amplitude scaling through learned filters and, when included, normalization layers. While not inherently part of CNNs, normalization layers are commonly used to stabilize training. For SVM and KNN, the feature extraction method is also scale invariant (we only consider position of the peaks).
- **Baseline variation:** CNNs, unlike traditional models, can learn to internally correct for baseline drift via convolutional filters. This eliminates the need for manual or rule-based preprocessing.
- **Peak shifts:** These represent the most challenging distortion. Slight shifts in Raman peak positions may result from changes in temperature, impurities, pressure, or instrument calibration. As peaks move, their associated representations change — possibly leading to misclassification. The feature extraction methods we presented in Section 2.2 for traditional ML classifiers are not invariant to shifts in peak positions. If a peak shifts enough to fall into a different interval, the features for that spectrum would change, impacting the classifier's performance. Convolutional-based models can account for translational invariance through the combination of convolutional and pooling layers. However, excessive invariance to translations can be detrimental. While smaller translations may be attributed to noise, higher translations may correspond to a different mineral. Hence, the classifier should be robust to minor translations but remain sensitive to significant ones. It is imperative to adjust the inductive bias of the CNNs to meet this requirement.

To better understand the behavior of CNNs in this context, we recall the concepts of equivariance and invariance:

Equivariance A transformation f is equivariant with respect to a group of action g if, for any g, the following relation holds:

$$f(g(x)) = g(f(x)). \tag{1}$$

In other words, if the input features are translated by a function g, then the output would also be translated by the same function. In the case of the convolution operation, defined as shown in equation 2, where x is the input signal and w is the convolution kernel, it is clear that a shift on x would lead to the same shift in $S(t)$.

$$S(t) = (x * w)(t) = \sum_n x(n)w(t-n). \tag{2}$$

Invariance In the case of an invariant transformation, the formulation would be:

$$f(g(x)) = f(x), \tag{3}$$

which means that a translation of the input features does not alter the output.

For our application, a fully equivariant response—exhibiting zero invariance—is not desirable. When a peak appears at a slightly

higher Raman shift, a purely equivariant system would produce a different representation, despite the underlying molecular identity remaining unchanged. In practice, such small variations often result from experimental noise or instrument calibration differences. Ideally, the model should learn representations that are invariant to these minor shifts. In convolutional neural networks, this invariance is introduced primarily through pooling layers. Specifically, max pooling is frequently used as a downsampling operation, which computes the maximum value over a window of size m applied to the output of the convolutional layer. This operation, illustrated in Figure 12, reduces the dimensionality of the feature maps while introducing local translational invariance. For Raman spectra, achieving an appropriate degree of translational invariance is essential: the model must be robust to small shifts caused by noise, yet still sensitive to larger shifts that may reflect different mineral species. Therefore, carefully tuning the pooling parameters is critical for adapting CNNs to Raman data classification.

To explore this, we conduct experiments with artificially shifted Raman spectra, using controlled displacements of ± 15 and $\pm 30\,\mathrm{cm}^{-1}$. Based on a manual inspection of the RRUFF dataset and reports that under harsh conditions, Raman spectra can exhibit shifts of up to $30\,\mathrm{cm}^{-1}$ (51), we adopt this value as a conservative upper bound on natural variation. An example of such a shifted sample is presented in Figure 13.

5.2 Experiments: Tuning CNNs for Partial Translational Invariance

We examine how CNNs behave under spectral shifts by varying two key architectural parameters:

- The pooling size m, which controls how much each pooling layer downscales the feature map.
- The number of pooling layers n, which affects how much invariance is accumulated across the network.

Larger m or higher n increases translational invariance — making the network more robust to shifts, but also potentially less discriminative.

We compare CNNs with low pooling size ($m = 2$) and high pooling size ($m = 64$), as well as networks with a shallow depth ($n = 1$) and a deeper configuration ($n = 10$). We report the Top-1 accuracy (fraction of spectra whose highest-probability prediction matches the label) and the Top-3 accuracy (fraction where the true class is among the three highest-probability predictions) for $m = 2$ and $m = 64$ in Table 3, and the Top-1 and the Top-3 accuracy for $n = 1$ and $n = 5$ in Table 4.

- CNNs with small pooling (low m or low n) perform well on clean data but are highly sensitive to shifts (Top-3 accuracy drops from 91% to 16%).
- CNNs with high pooling parameters show significantly better robustness (Top-3 accuracy of 57% under $30\,\mathrm{cm}^{-1}$ shift), albeit at a small cost in unshifted accuracy.

We further examine Top-3 predictions in Tables 5 and 6. CNNs with higher pooling correctly retain the true class (PyrosmaliteFe) among the Top-3 predictions for shifted inputs, while models with low pooling completely fail.

Table 5 represents the Top-3 predictions made by a model consisting of 1 and 10 convolutional blocks, both for the original sample and the shifted one. The model with only 1 block predicts completely different minerals in the case of the shifted sample, while the model with 10 blocks is able to account for the shift and the Top-3 predictions are the same. Table 6 represent the predictions for a model where the m parameter takes values $m = 2$ and $m = 64$ respectively. The results obtained are consistent with the previous experiments, demonstrating that tuning m can be considered as a hyperparameter that controls the degree of translational invariance. In particular, it can be tuned considering the particular instrument that is going to

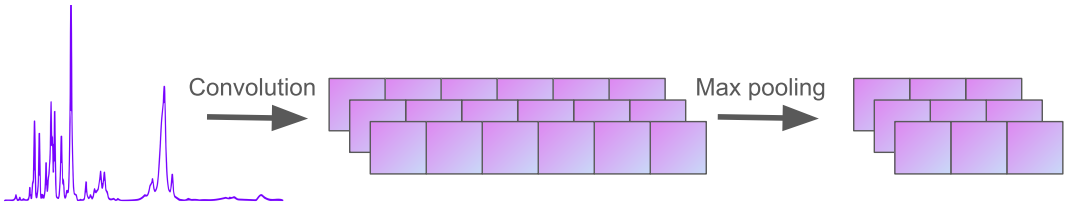


Figure 12: Schematic of the max-pooling operation in a CNNs for Raman spectral analysis. The convolutional layers extract hierarchical features from the input Raman spectrum, while max pooling reduces the spatial dimensions of the feature maps, introducing local translational invariance and enhancing robustness to spectral shifts.

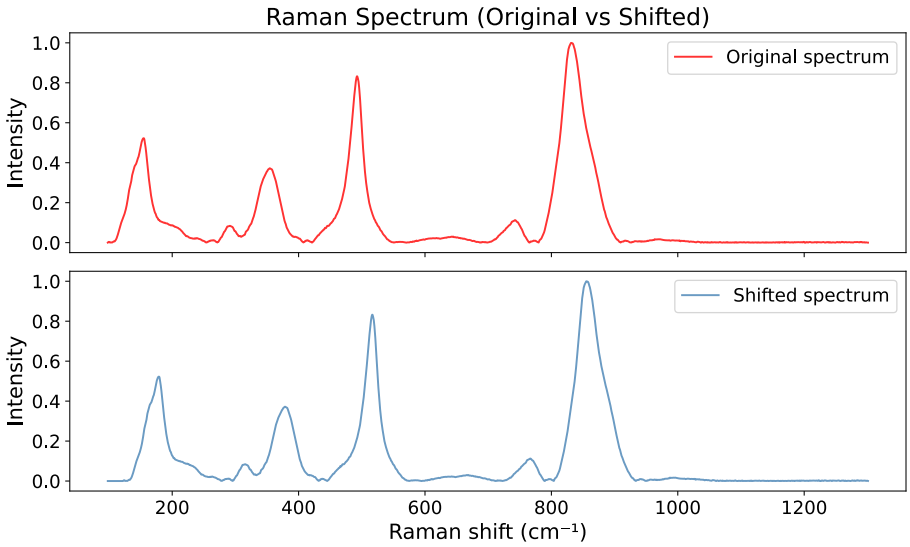


Figure 13: Comparison of the original and shifted Raman spectra. The top panel shows the original spectrum, while the bottom panel displays the spectrum shifted by 30 cm^{-1}.

be used, knowing the amount of shift that can be attributed to the experimental conditions. Nonetheless, excessive pooling results in a loss of spatial information, limiting the model's capacity to differentiate between closely related spectra.

These experiments confirm that CNNs pooling parameters (m, n) act as knobs for controlling translational invariance, allowing practitioners to tailor CNNs behavior to the known variance of their spectroscopic instrumentation. This finding has direct implications for practitioners: pooling parameters can be adjusted based on the expected instrumental variance, allowing models to be tailored for specific experimental setups.

6 Classification using semi-supervised methods

CNNs typically require large amounts of labeled data, which is often expensive and time-consuming to obtain for Raman spectroscopy. In this section, we investigate two semi-supervised learning strategies to address this limitation: semi-supervised generative adversarial networks (SGAN) (52) (53) and contrastive learning (54). Both methods aim to improve classification performance by leveraging unlabeled data.

6.1 Methodology

Semi-supervised generative adversarial networks (SGAN) Semi-supervised gen-

Table 3: Top-1 and Top-3 accuracy on unshifted and shifted samples, using $m = 2$ and $m = 64$. Larger pooling sizes ($m = 64$) show improved robustness to spectral shifts but slightly reduced accuracy on unshifted samples, reflecting a trade-off between translational invariance and resolution.

Pooling parameter	$m = 2$		$m = 64$	
Accuracy (%)	Top-1	Top-3	Top-1	Top-3
No shift	83	91	62	82
Shifted by $15\,\mathrm{cm}^{-1}$	29	51	47	65
Shifted by $30\,\mathrm{cm}^{-1}$	8	16	39	57

Table 4: Top-1 and Top-3 accuracy on unshifted and shifted samples, using $n = 1$ and $n = 10$ pooling layers. Increasing the number of pooling layers ($n = 10$) shows improved robustness to spectral shifts but slightly reduced accuracy on unshifted samples, emphasizing the importance of tuning inductive bias.

Number of pooling layers	n=1		n=10	
Accuracy (%)	Top-1	Top-3	Top-1	Top-3
No shift	83	91	63	80
Shifted by $15\,\mathrm{cm}^{-1}$	29	51	42	70
Shifted by $30\,\mathrm{cm}^{-1}$	6	15	46	63

erative adversarial networks (SGAN) build upon the generative adversarial network (GAN) framework, which involves two neural networks—a generator and a discriminator—trained in an adversarial setting. In the GAN framework, the generator synthesizes data that mimics the real distribution, while the discriminator learns to distinguish between real and synthetic samples. SGANs extend this framework by modifying the discriminator to output a probability distribution over $N + 1$ classes—one for each of the N real classes, plus an additional class corresponding to generated (synthetic) data. This allows the discriminator to simultaneously perform classification and real/synthetic discrimination. As a result, SGANs can leverage both labeled and unlabeled data during training, improving classification performance in low-supervision settings (see Supporting Information for details).

In our implementation, both generator and discriminator use 1-D CNNs suited for spectral data. The generator starts from a latent vector of dimension 128 and upscales it using transposed convolutions to match the spectral shape. The discriminator uses the same base CNNs as our earlier supervised classifier (Figure 6), with an additional output head for binary discrimination (Figure 14).

Contrastive learning Contrastive learning trains models to distinguish between similar and dissimilar data points by bringing similar pairs closer together in the representation space while pushing dissimilar pairs further apart. This is achieved by maximizing the agreement between augmented views of the same sample. We adapt the SimCLR framework (54) for Raman spectra using 1-D CNNs (Figure 15).

Table 5: Top-3 predicted mineral classes for a PyrosmaliteFe sample under non-shifted and shifted conditions ($30\,\mathrm{cm}^{-1}$) with $n = 1$ and $n = 10$ pooling layers. Higher pooling depth preserves correct predictions under spectral shifts.

# of pooling layers	$n = 1$			$n = 10$		
Predicted class	Top-1	Top-2	Top-3	Top-1	Top-2	Top-3
No shift	PyrosmaliteFe	Rutile	Diopside	PyrosmaliteFe	Magnetite	Grunerite
Shifted by $30\,\mathrm{cm}^{-1}$	Augelite	Pectolite	Lazulite	PyrosmaliteFe	Grunerite	Magnetite

Table 6: Top-3 predicted mineral classes for a PyrosmaliteFe sample under non-shifted and shifted conditions ($30\,\mathrm{cm}^{-1}$) with pooling sizes $m = 2$ and $m = 64$. Larger pooling improves robustness to spectral shifts, maintaining the correct class among the Top-3 predictions.

Pooling parameter	$m = 2$			$m = 64$		
Predicted class	Top-1	Top-2	Top-3	Top-1	Top-2	Top-3
No shift	PyrosmaliteFe	Orthoserpierite	Paravauxite	PyrosmaliteFe	Orthoserpierite	Paravauxite
Shifted by $30\,\mathrm{cm}^{-1}$	Grunerite	Clintonite	Beryl	PyrosmaliteFe	Paravauxite	Beryl

Given a raw spectrum, we generate two stochastic augmentations using domain-specific transformations that preserve peak positions. The most general transformation we use is adding a function, such as $\tanh(\cdot)$ or a $\cos(\cdot)$ functions, with the magnitude scaled by the mean value of the spectrum. Adding a $\cos(\cdot)$ function, despite potentially introducing a new peak, can mimic the noise induced by fluorescence. Another transformation is the addition of Gaussian noise, simulating measurements from different instruments. Lastly, down-sampling and interpolating the signal, with a random selection of points in the spectrum, is used. The interpolation parameters increase the stochasticity of the method, resulting in a higher variance in the augmented spectrum. A higher weight is used for this method (that is, interpolation is used more frequently). As noted by Cheng *et al.*(54), unsupervised contrastive learning benefits more from data augmentation than supervised approaches, but it is crucial to ensure that the transformations cover the intra-class variance for each mineral. The encoder $f(\cdot)$ maps augmented spectra to latent features, followed by a projection head $g(\cdot)$. After unsupervised pre-training, $g(\cdot)$ is discarded, and a classifier is trained on top of frozen $f(\cdot)$ using the labeled samples (see Supporting Information for details).

6.2 Experiments

To simulate low-resource settings, we artificially mask the labels of a portion of the dataset, creating supervised subsets with 10% to 50% labeled samples. We evaluate SGAN and Contrastive learning on the raw Raman spectra and compare them to a baseline CNNs trained solely on the labeled subset.

SGAN: Qualitative Evaluation Figure 16 compares a real Perovskite spectrum with a synthetic one generated by the SGAN. While both spectra exhibit a similar overall shape, there are noticeable differences, particularly in the noise pattern. For instance, although both spectra show a prominent peak, the peak in the generated spectrum is shifted and exhibits a distinct noise pattern compared to the real one. The primary goal is not to generate perfectly realistic samples but to enhance the per-

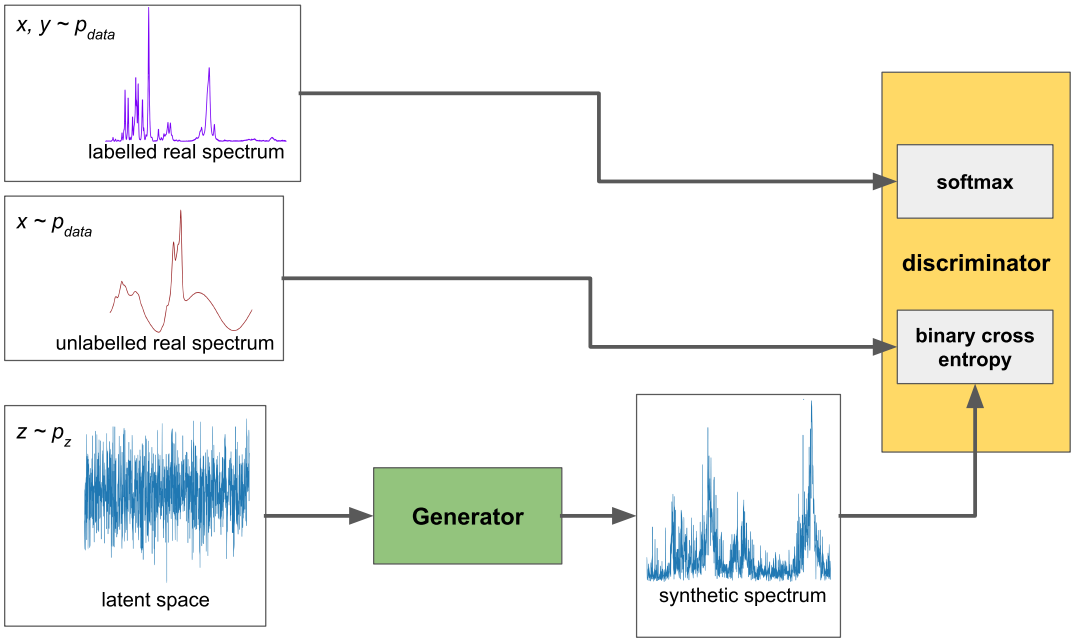


Figure 14: The architecture of the SGAN applied to Raman data, featuring a 1-D CNNs-based generator that upscales latent space representations to match real spectrum dimensions, and a discriminator with an additional binary classification output to distinguish real from synthetic spectra.

formance of the discriminator/classifier through semi-supervised learning. If the generator produces samples that perfectly match the real data distribution $p(\mathbf{X})$, the decision boundaries of the discriminator remain unchanged. On the other hand, if the generated samples are *complementary* to $p(\mathbf{X})$ in the feature space, this will help to obtain better decision boundaries (55). Therefore, having generated samples that do not look exactly like the real ones is desirable in the case of SGAN, unlike traditional GAN where the principal purpose is to generate realistic samples.

The generator's task is not just to mimic real data but to create variations in the data that contribute to a richer representation of the feature space, which aids in better decision-making by the discriminator. This process can be thought of as generating additional, diverse data that helps the classifier learn more discriminative features, similar to how data augmentation methods like adding Gaussian noise can introduce variability. In both cases, the goal is to increase the robustness of the model by forcing it to adapt to these new, challenging data points.

SGAN: Quantitative Results Table 7 shows classification accuracy on the test set using SGAN vs. a purely supervised CNNs. SGAN consistently outperforms the supervised baseline, especially when only 10%–30% of the data is labeled. For instance, at 10% labeled data, SGAN improves accuracy from 42% to 53%. The gain diminishes as more labels become available.

However, the improvement is less than anticipated. Despite using 90% unlabeled samples in addition to 10% labeled ones, the expected increase in accuracy is not as substantial. This behavior can be explained by the nature of Raman spectra. Most of the literature has focused on using semi-supervised approaches like SGAN to benefit the classification problem on images. In contrast, Raman spectra exhibit different characteristics compared to image data, with low-level features having limited complexity as discussed in Section 7. This can explain why using semi-supervised approaches in Raman spectra, although increasing the classifica-

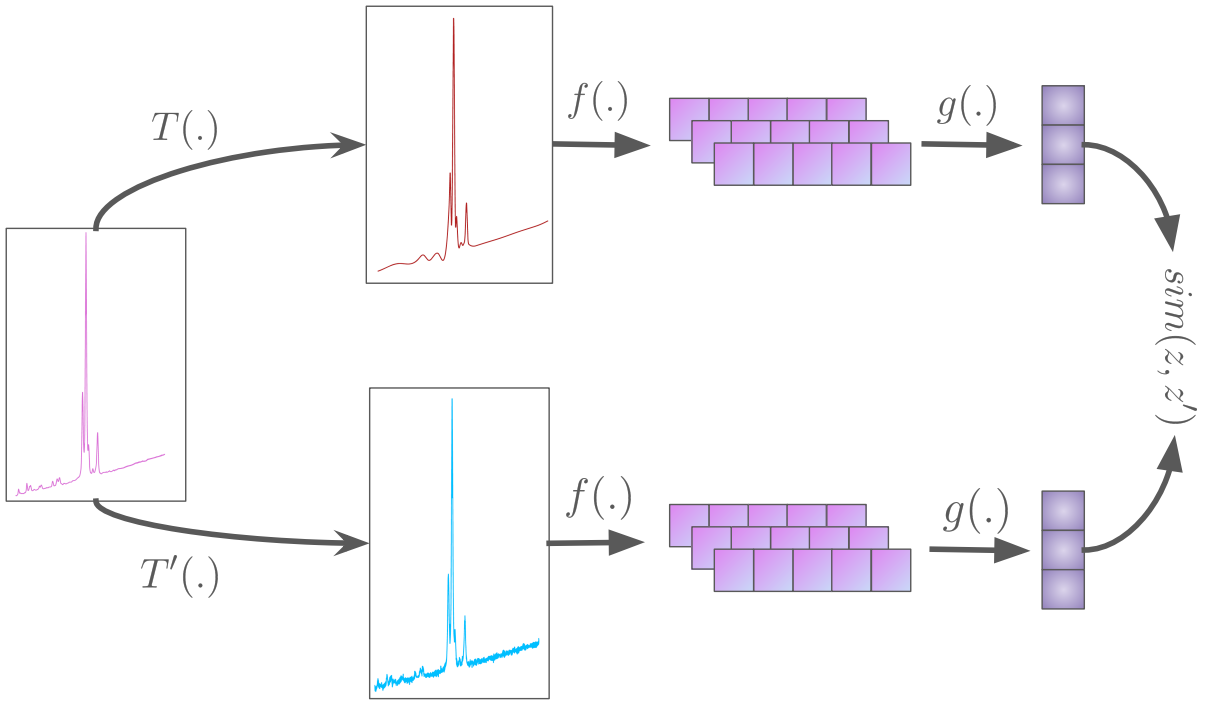


Figure 15: Unsupervised contrastive learning framework applied to Raman data. The raw Raman spectrum is transformed through data augmentation functions $T(\cdot)$ and $T'(\cdot)$, which generate two different augmented views of the same sample. These augmented spectra are encoded into feature representations by the 1-D CNNs encoder $f(\cdot)$. The encoded features are then passed through a projection head $g(\cdot)$ to produce latent vectors z and z'. Finally, the similarity $\text{sim}(z, z')$ between the two latent representations is computed, encouraging the model to maximize agreement between augmented views of the same spectrum.

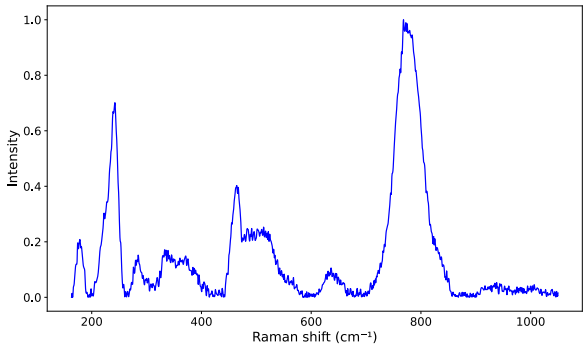
tion accuracy, has limited benefit compared to the computer-vision literature.

Contrastive learning: Quantitative Results Contrastive learning yields similar trends (Table 8). With only 10% labeled data, accuracy improves from 42% (supervised) to 51% (semi-supervised). Gains diminish with more labeled data, converging to the supervised baseline. Notably, pre-training provides robust representations even when labeled examples are scarce.

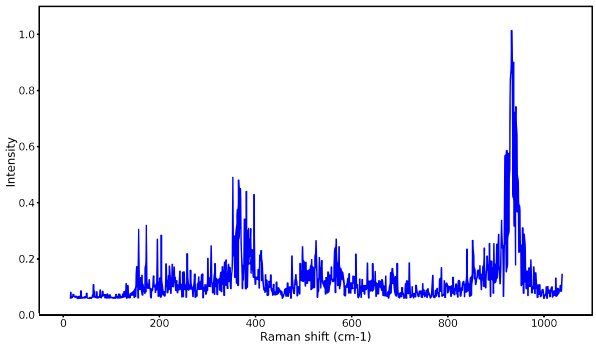
In the contrastive learning framework, the pre-training phase is effective for representation learning due to the inherent characteristics of the data. Raman spectra exhibit universal features that are independent of specific classes. For instance, the fluorescence baseline is a common element in nearly all raw sample, thus a model can learn an embedding of the spectrum that accounts for this phenomenon. More broadly, similarities across all Raman spectra arise from underlying physical phenomena. This process is analogous to how Language Models learn underlying syntactic and semantic rules or patterns when pretrained with an unsupervised task (56).

7 Assessing the learnability and representational efficiency of low-level features in Raman Spectra

In this section, we investigate the nature and complexity of low-level features in Raman spectra. In image classification, convolutional neural networks (CNNs) extract hierarchical representations, where early layers capture simple



(a) The real Perovskite Raman spectrum, showing characteristic peaks corresponding to the material's unique vibrational modes.



(b) The synthetic Raman spectrum generated by the SGAN's generator, mimicking the general structure and noise profile of the real spectrum while preserving key spectral features.

Figure 16: Comparison of real and synthetic Perovskite Raman spectra.

Table 7: Performance of fully supervised CNNs and SGAN approaches on the test set across varying proportions of labeled samples. The results show that the SGAN outperforms the fully supervised CNNs, particularly in scenarios with fewer labeled samples, highlighting the advantage of semi-supervised learning in leveraging unlabeled data.

Proportion of labeled samples (%)	10	20	30	40	50
Accuracy with only Supervised training (%)	42	58	65	69	76
Accuracy with Semi-Supervised training (%)	53	66	67	71	77

features like edges, and deeper layers learn increasingly abstract patterns (27). While this hierarchical structure has proven effective in computer vision, it is not clear whether Raman spectra exhibit similarly complex low-level patterns.

We hypothesize that the low-level structure of Raman data enables efficient learning, such that meaningful features can be extracted from a limited number of examples. This could explain the limited gains observed with semi-supervised learning in Section 6. To test this hypothesis, we designed two complementary experiments: first, we assess the sample efficiency of low-level feature learning using partial layer freezing in a CNNs; second, we examine the general compressibility of Raman spectra by learning sparse latent representations in an unsupervised setting. Together, these experiments allow us to evaluate both the complexity and the learnability of low-level spectral features, from both supervised and unsupervised perspectives.

7.1 Methodology

We structured our investigation into two targeted experiments:

1. **Experiment 1: Layer Freezing** We first focus on assessing how many labeled examples are required to learn meaningful low-level features in a supervised setting. Specifically, we train the early layers of a CNNs (first two layers) using small subsets of the data, freeze them — meaning we stop updating their weights during further training — and subsequently train the remaining layers on the full dataset. Comparing this approach to an end-to-

Table 8: Performance of contrastive semi-supervised learning and fully supervised CNNs approaches on the test set across different proportions of labeled samples. The semi-supervised contrastive learning approach consistently improves classification accuracy, demonstrating its ability to effectively utilize unlabeled data to enhance performance.

Proportion of labeled samples (%)	10	20	30	40	50
Accuracy with only Supervised training (%)	42	58	65	69	76
Accuracy with Semi-Supervised training (%)	51	65	69	75	76

end trained CNNs helps us quantify the sample efficiency of early feature learning.

2. **Experiment 2: Sparse Autoencoder** To complement the supervised perspective of Experiment 1, we explore whether Raman spectra can be effectively compressed into sparse latent representations in an unsupervised way. Here, we train a fully connected autoencoder to minimize reconstruction error, encouraging the model to capture the overall structure of the spectra without supervision or convolutional priors. The learned latent representation is then used as input to a downstream classifier, allowing us to evaluate how well the unsupervised features transfer to the classification task.

7.2 Experiments

Experiment 1: Layer Freezing This experiment evaluates whether low-level features can be learned effectively from a limited subset of labeled data. We train the first two layers of a CNNs using subsets of increasing size (80, 200, and 848 samples), freeze them, and then train the remaining layers on the full dataset. As shown in Table 9, performance remains stable between 200 and 848 samples, but drops from 83 % to 76 % with only 80 samples. These results indicate that low-level features in Raman spectra can be reliably learned from relatively few examples, highlighting the favorable sample efficiency of the data.

Experiment 2: Sparse Autoencoder While Experiment 1 demonstrates the sample efficiency of supervised learning, it does not address whether Raman spectra are inherently compressible in an unsupervised context. To investigate this, we train a fully connected autoencoder using a reconstruction loss (Equation 4). The choice of reconstruction loss is critical, as it enforces the learning of a comprehensive representation of the input spectra, capturing both class-relevant and class-agnostic features. Unlike classification loss, which prioritizes discriminative characteristics, reconstruction loss ensures that the latent space encodes the full spectral information, independently of the class labels.

In addition, by employing fully connected layers, we deliberately avoid convolutional inductive biases, allowing us to evaluate the capacity of dense architectures to model the underlying structure of Raman spectra. Following the unsupervised pretraining phase, the autoencoder layers are frozen, and the latent representations $\mathbf{h}$ are used as input features for a downstream classifier. This approach enables us to assess whether the unsupervised representations preserve sufficient discriminative information for accurate classification. To ensure methodological rigor, the dataset used for autoencoder pretraining was excluded from the subsequent classifier training. The complete experimental workflow is depicted in Figure 17, illustrating the process of unsupervised autoencoder pretraining, extraction of latent spectral features, and their utilization in downstream classification.

Table 9: CNNs accuracy when pretraining the first two layers on subsets of increasing size. Results indicate that meaningful low-level features in Raman spectra can be learned efficiently from limited labeled data, highlighting favorable sample efficiency.

Training Samples for Layers 1-2	80	200	848
Test Accuracy (%)	76	82	83

$$L(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{N} (\mathbf{y}_i - \hat{\mathbf{y}}_i(\mathbf{x}))^2 = (\mathbf{y}_i - \mathrm{MLP}(\mathbf{x}_i))^2 \,. \tag{4}$$

The results of the autoencoder experiment are presented in Table 10. Despite the absence of convolutional structures and supervised training signals, the autoencoder successfully learns sparse latent representations that enable the downstream classifier to reach competitive performance. This finding complements our observations from Experiment 1: while early CNNs layers require only limited data to extract meaningful features, the autoencoder further demonstrates that Raman spectra possess an intrinsic structure that can be captured in an unsupervised and non-convolutional framework.

Notably, these results are in agreement with our previous feature engineering approach, where manually identified peak positions provided valuable classification cues (Section 3). Here, the autoencoder extracts comparable low-level information directly from raw spectral inputs, without explicit peak detection or prior assumptions about spectral features. These findings reinforce the notion that the low-level representations in Raman spectra are both compact and discriminative, and they further support the development of efficient modeling strategies leveraging sparse representations for downstream tasks.

8 Classification using transfer learning

Building on the insights gained from semi-supervised learning and sparse autoencoder experiments, we now explore whether representations learned by neural networks can generalize to previously unseen classes. The earlier sections demonstrated that CNNs are capable of learning robust, low-level features directly from raw Raman spectra, and that these features are sufficiently simple to be captured from a limited number of samples. These findings naturally raise the following question: can we reuse these learned features to recognize new mineral classes, without retraining the entire model from scratch?

In previous work, Liu *et al.* (24) identified a limitation of CNNs-based models for Raman classification: the need to fully retrain the network when new classes are introduced. To address this, they proposed a Siamese network architecture, which compares input spectra to reference samples. However, Siamese networks suffer from increasing inference time and storage requirements as the number of reference classes grows, since each new sample must be compared to all stored references.

As an alternative, we investigate **transfer learning**, which allows a pretrained model to be adapted to new classes with minimal retraining. This approach promises constant inference time, as new classes can be accommodated simply by retraining the final classification layer, while keeping the feature extractor frozen. Given the favorable sample efficiency of Raman spectral features and their consistency across mineral families (as shown in Section 7), we hypothesize that transfer learning can efficiently extend the model to new classes. Compared to Siamese networks, which require $\mathcal{O}(k)$ comparisons to reference classes at inference time, transfer learning reduces the complexity to $\mathcal{O}(1)$, enabling constant inference time regardless of the number of known classes.

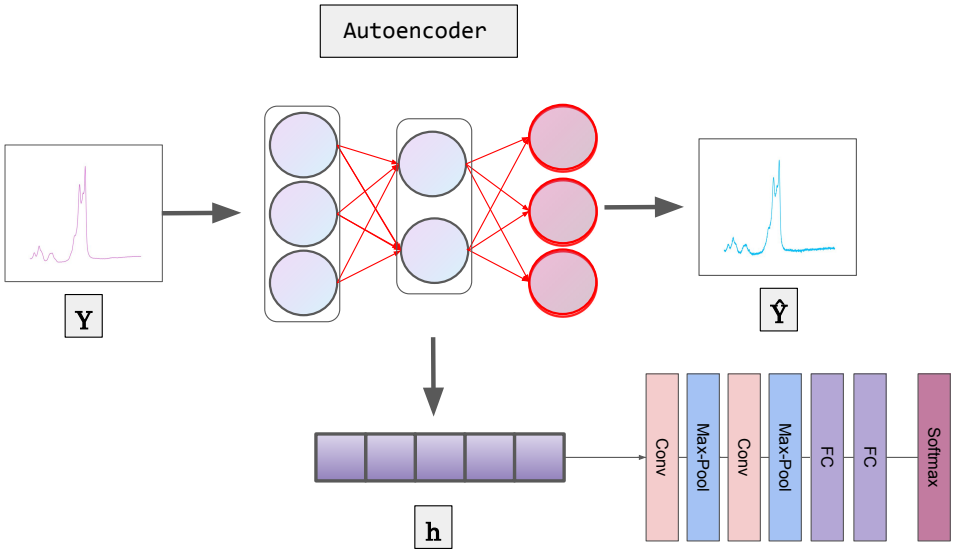


Figure 17: Autoencoder experiment pipeline. The autoencoder is pretrained on a sparse reconstruction task to minimize the reconstruction error $L(\mathbf{y}, \hat{\mathbf{y}})$, where $\mathbf{y}$ is the original spectrum and $\hat{\mathbf{y}}$ is the reconstructed output. This unsupervised step encourages the model to capture the full underlying structure of the spectra, independently of class labels. Low-level features are encoded in the latent representation $\mathbf{h}$ ($\dim(\mathbf{h}) = 256$). After freezing the autoencoder, $\mathbf{h}$ is used as input to train a classifier for the classification task, with the subset of data used for pretraining excluded from fine-tuning.

Table 10: Comparison of fully supervised training and autoencoder-based feature extraction across varying proportions of labeled samples. The autoencoder captures informative representations of Raman spectra in an unsupervised manner, enabling competitive classification accuracy despite limited supervision.

Proportion of labeled samples (%)	10	20	30	40	50
Accuracy with fully supervised CNNs (%)	42	58	65	69	76
Accuracy using CNNs with autoencoder features (%)	43	53	58	66	67

8.1 Methodology

To evaluate the potential of transfer learning for Raman spectral classification, we design the following experiment:

- Split the full dataset of n classes into two subsets: $n - c$ classes for initial training and c classes reserved for fine-tuning.
- Train a CNNs-based model on the $n - c$ classes, and freeze all layers of the network — that is, we stop updating their weights — except for the classification head.
- Replace the classification layer to accommodate the c new classes, and fine-tune only this layer for a small number of epochs.

This protocol tests whether the features learned during initial training are sufficiently general to transfer effectively to new mineral classes, without retraining the convolutional backbone.

8.2 Experiments

The results of this experiment are summarized in Table 11. When the number of new classes c

Table 11: Transfer learning results for Raman spectral classification across increasing numbers of new classes (c). High performance is maintained for small c, but declines as c increases, reflecting the combined effects of reduced feature extractor diversity and increased classification complexity.

	$c = 5$	$c = 10$	$c = 15$	$c = 20$
Accuracy on $n - c$ classes (pretraining) (%)	85	80	84	80
Accuracy on c classes (fine-tuning) (%)	89	78	63	43

is small relative to the total number of classes n, transfer learning achieves classification accuracy comparable to models trained from scratch on these new classes. However, as c increases, we simultaneously reduce the number of pretraining classes $n - c$, which limits the diversity of the feature extractor, and increase the number of target classes, which makes the classification task inherently more complex. This combined effect leads to a degradation in performance, as shown in Table 11.

The training and validation curves during the fine-tuning phase for different values of c are shown in Supporting Information, Figures S17-S20. Despite using a conservative learning rate of $\eta = 10^{-5}$ to mitigate catastrophic forgetting, the fine-tuning process quickly converges. For small c, convergence is fast and performance remains high, consistent with our findings in Section 7 that the learned feature space is generalizable. However, as c increases, the reduced expressiveness of the frozen feature space becomes a limiting factor, and the classifier struggles to accommodate the higher number of target classes.

These results suggest that the success of transfer learning for Raman data is underpinned by the shared structural and compositional characteristics of minerals within the same family (in this context family can be substituted by another concept in the taxonomic hierarchy). Even if the model has not seen a specific mineral class during training, it can still classify it correctly if it has been exposed to related classes. This observation is coherent with our findings from the autoencoder experiment (Section 7), which demonstrated that low-level features generalize well across the dataset.

9 Conclusion

Raman spectroscopy is now deployed from planetary missions on Mars (57, 58) to the investigation of hydrothermal vents on Earth (59, 60), and is even envisioned to facilitate the search for life on other worlds (61). Across all of these settings, spectral interpretation faces the same core challenges: limited bandwidth, variable environmental conditions, and the need for reliable onboard autonomy. To address these challenges, we present a deployable, data-efficient deep learning framework that enables four key capabilities:

1. **Baseline-free accuracy** 1-D CNNs surpass k-nearest-neighbors and support-vector classifiers built on handcrafted peak features, eliminating background correction and peak picking. We also introduce physically plausible data augmentation strategies—such as controlled noise and amplitude scaling—that preserve Raman peak positions while improving generalization. All splits and preprocessing scripts are released to ensure full reproducibility.

2. **A one-knob robustness dial** Tuning a single pooling parameter allows CNNs to absorb Raman shift displacements up to 30 cm^{-1} without degrading class resolution—providing a principled way to align model behavior with instrument stability and expected variability.

3. **Learning from unsupervised data** Semi-supervised GANs and contrastive pretraining raise accuracy by up to 11 % with only 10 % of labels. Gains are modest—yet operationally crucial—due to the

intrinsic information density of Raman spectra, as confirmed by inductive-bias experiments.

4. **Constant-time transfer** Freezing the CNNs backbone and re-training a softmax head adapts the model to unseen minerals at $\mathcal{O}(1)$ inference cost, outperforming Siamese and reference-matching designs on embedded systems.

Reproducibility. All dataset splits are publicly archived at `https://github.com/denizsoysal/Raman_spectra_data` to serve as a benchmark for future studies.

Outlook. As open spectral libraries expand and instrument-specific distortions become better characterized, the proposed framework—training directly on raw spectra, tuning pooling to match hardware drift, leveraging semi-supervised learning, and fine-tuning lightly for new targets—offers a scalable foundation for general-purpose Raman classifiers across minerals, organics, and biomaterials. These tools can help unlock the full potential of autonomous chemistry missions across planetary and oceanographic frontiers.

Supporting Information

Introduction to Raman spectroscopy, detailed descriptions of the machine learning models used in this study including support vector machines, k-nearest neighbors, and convolutional neural networks, training curves and additional information on the semi-supervised and contrastive learning methods (.PDF format).

Acknowledgements

Deniz Soysal and Xabier García–Andrade conducted this research as part of their graduate research at KU Leuven. Renaud Detry was supported by Interne Fondsen KU Leuven/Internal Funds KU Leuven. Laura E. Rodriguez was supported by the Lunar and Planetary Institute operated by the Universities Space Research Association (LPI contribution XXXX). Pablo Sobron was supported by the SETI Institute and Impossible Sensing. Work by Laura M. Barge was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA (80NM0018D004). Government sponsorship acknowledged.

References

(1) Long, T.; Zhou, Z.; Hancke, G.; Bai, Y.; Gao, Q. A Review of Artificial Intelligence Technologies in Mineral Identification: Classification and Visualization. *J. Sens. Actuat. Netw.* **2022**, *11*, 50.

(2) Pomyen, Y.; Wanichthanarak, K.; Poungsombat, P.; Fahrmann, J.; Grapov, D.; Khoomrung, S. Deep Metabolome: Applications of Deep Learning in Metabolomics. *Comput. Struct. Biotechnol. J.* **2020**, *18*, 2818–2825.

(3) Signoroni, A.; Savardi, M.; Baronio, A.; Benini, S. Deep Learning Meets Hyperspectral Image Analysis: A Multidisciplinary Review. *J. Imaging* **2019**, *5*, 52.

(4) Chen, D.; Wang, Z.; Guo, D.; Orekhov, V.; Qu, X. Review and Prospect: Deep Learning in Nuclear Magnetic Resonance Spectroscopy. *Chem. Eur. J.* **2020**, *26*, 10391–10401.

(5) Mamede, R.; Pereira, F.; Aires-de Sousa, J. Machine Learning Prediction of UV–Vis Spectra Features of Organic Compounds Related to Photoreactive Potential. *Sci. Rep.* **2021**, *11*, 23720.

(6) Sadaiappan, B.; Balakrishnan, P.; C.R., V.; Vijayan, N. T.; Subramanian, M.; Gauns, M. U. Applications of Machine Learning in Chemical and Biological Oceanography. *ACS Omega* **2023**, *8*, 15831–15853.

(7) Debus, B.; Parastar, H.; Harrington, P.; Kirsanov, D. Deep Learning in Analytical Chemistry. *TrAC Trends Anal. Chem.* **2021**, *145*, 116459.

(8) Ryzhikova, E.; Ralbovsky, N. M.; Sikirzhytski, V.; Kazakov, O.; Halamkova, L.; Quinn, J.; Zimmerman, E. A.; Lednev, I. K. Raman Spectroscopy and Machine Learning for Biomedical Applications: Alzheimer's Disease Diagnosis Based on the Analysis of Cerebrospinal Fluid. *Spectrochim. Acta Part A* **2021**, *248*, 119188.

(9) Banas, A. M.; Banas, K.; Breese, M. B. H. Classification of the Residues after High and Low Order Explosions Using Machine Learning Techniques on Fourier Transform Infrared (FTIR) Spectra. *Molecules* **2023**, *28*, 2233.

(10) Lei, B.; Bissonnette, J. R.; Hogan, U. E.; Bec, A. E.; Feng, X.; Smith, R. D. L. Customizable Machine-Learning Models for Rapid Microplastic Identification Using Raman Microscopy. *Anal. Chem.* **2022**, *94*, 17011–17019.

(11) Barker, L. D. L.; Jakuba, M. V.; Bowen, A. D.; German, C. R.; Maksym, T.; Mayer, L.; Boetius, A.; Dutrieux, P.; Whitcomb, L. L. Scientific Challenges and Present Capabilities in Underwater Robotic Vehicle Design and Navigation for Oceanographic Exploration Under-Ice. *Remote Sensing* **2020**, *12*, 2588.

(12) Lawrence, J. D. et al. Crevasse Refreezing and Signatures of Retreat Observed at Kamb Ice Stream Grounding Zone. *Nat. Geosci.* **2023**, *16*, 238–243.

(13) Liu, Q.; Guo, J.; Lu, Y.; Wei, Z.; Liu, S.; Wu, L.; Ye, W.; Zheng, R.; Zhang, X. Underwater Raman Microscopy—A Novel in Situ Tool for Deep-Sea Microscale Target Studies. *Front. Mar. Sci.* **2022**,

(14) White, S.; Dunk, R.; Peltzer, E.; Freeman, J.; Brewer, P. In Situ Raman Analyses of Deep-Sea Hydrothermal and Cold Seep Systems. *Geochem. Geophys. Geosyst.* **2006**, *7*.

(15) Takahashi, T.; Yoshino, S.; Takaya, Y.; Nozaki, T.; Ohki, K.; Ohki, T.; Sakka, T.; Thornton, B. Quantitative in Situ Mapping of Elements in Deep-Sea Hydrothermal Vents Using Laser-Induced Breakdown Spectroscopy and Multivariate Analysis. *Deep Sea Res. Part I* **2020**, *158*, 103232.

(16) McMutrtry, G. M.; Wiltshire, J. C.; Bossuyt, A. *Europe Oceans 2005*; 2005; Vol. 1; pp 395–400.

(17) Zhang, Y.; Ryan, J. P.; Kieft, B.; Hobson, B. W.; McEwen, R. S.; Godin, M. A.; Harvey, J. B.; Barone, B.; Bellingham, J. G.; Birch, J. M.; Scholin, C. A.; Chavez, F. P. Targeted Sampling by Autonomous Underwater Vehicles. *Front. Mar. Sci.* **2019**,

(18) Theiling, B. P.; Chou, L.; Poian, V.; Battler, M.; Raimalwala, K.; Arevalo, R.; Neveu, M.; Ni, Z.; Graham, H.; Elsila, J.; Thompson, B. Science Autonomy for Ocean Worlds Astrobiology: A Perspective. *Astrobiology* **2022**, *22*, 901–913.

(19) Poian, V.; Lyness, E.; Danell, R.; Li, X.; Theiling, B.; Trainer, M.; Kaplan, D.; Brinckerhoff, W. Science Autonomy and Space Science: Application to the ExoMars Mission. *Front. Astron. Space Sci.* **2022**,

(20) Sartore, C.; Simetti, E.; Wanderlingh, F.; Casalino, G. Autonomous Deep Sea Mining Exploration: The EU ROBUST Project Control Framework. OCEANS 2019. 2019; pp 1–8.

(21) Wettergreen, D.; Foil, G.; Furlong, M.; Thompson, D. R. Science Autonomy for Rover Subsurface Exploration of the Atacama Desert. *AI Mag.* **2014**, *35*, 47–60.

(22) Hand, K. P.; German, C. R. Exploring Ocean Worlds on Earth and Beyond. *Nat. Geosci.* **2018**, *11*, 2–4.

(23) Liu, J.; Osadchy, M.; Ashton, L.; Foster, M.; Solomon, C. J.; Gibson, S. J.

Deep Convolutional Neural Networks for Raman Spectrum Recognition: A Unified Solution. *Analyst* **2017**, *142*, 4067–4074.

(24) Liu, J.; Gibson, S. J.; Mills, J.; Osadchy, M. Dynamic Spectrum Matching with One-Shot Learning. *Chemom. Intell. Lab. Syst.* **2019**, *184*, 175–181.

(25) Wiens, R. C. et al. Pre-Flight Calibration and Initial Data Processing for the ChemCam Laser-Induced Breakdown Spectroscopy Instrument on the Mars Science Laboratory Rover. *Spectrochim. Acta Part B* **2013**, *82*, 1–27.

(26) Kensert, A.; Collaerts, G.; Efthymiadis, K.; Broeck, P.; Desmet, G.; Cabooter, D. Deep Convolutional Autoencoder for the Simultaneous Removal of Baseline Noise and Baseline Drift in Chromatograms. *J. Chromatogr. A* **2021**, 462093.

(27) LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436.

(28) Berlanga, G.; Williams, Q.; Temiquel, N. Convolutional Neural Networks as a Tool for Raman Spectral Mineral Classification Under Low Signal, Dusty Mars Conditions. *Earth Space Sci.* **2022**, *9*, e2021EA002125.

(29) Xu, X.; Ma, F.; Zhou, J.; Du, C. Applying Convolutional Neural Networks for End-to-End Soil Analysis Based on Laser-Induced Breakdown Spectroscopy with Less Spectral Preprocessing. *Comput. Electron. Agric.* **2022**, *199*, 107171.

(30) Hou, X.; Wang, G.; Wang, X.; Ge, X.; Fan, Y.; Nie, S. Convolutional Neural Network-Based Approach for Classification of Edible Oils Using Low-Field Nuclear Magnetic Resonance. *J. Food Compos. Anal.* **2020**, *92*, 103566.

(31) Shariat, K.; Kirsanov, D.; Olivieri, A. C.; Parastar, H. Sensitivity and Generalized Analytical Sensitivity Expressions for Quantitative Analysis Using Convolutional Neural Networks. *Anal. Chim. Acta* **2022**, *1192*, 338697.

(32) Zhang, R.; Xie, H.; Cai, S.; Hu, Y.; Liu, G.-K.; Hong, W.; Tian, Z.-q. Transfer-Learning-Based Raman Spectra Identification. *J. Raman Spectrosc.* **2019**, *51*, 176–186.

(33) Blazhko, U.; Shapaval, V.; Kovalev, V.; Kohler, A. Comparison of Augmentation and Pre-Processing for Deep Learning and Chemometric Classification of Infrared Spectra. *Chemom. Intell. Lab. Syst.* **2021**, *215*, 104367.

(34) Wang, W.; Liu, X.; Mou, X. Data Augmentation and Spectral Structure Features for Limited Samples Hyperspectral Classification. *Remote Sensing* **2021**, *13*, 547.

(35) van Engelen, J. E.; Hoos, H. H. A Survey on Semi-Supervised Learning. *Mach. Learn.* **2020**, *109*, 373–440.

(36) Wu, H.; Prasad, S. Semi-Supervised Deep Learning Using Pseudo Labels for Hyperspectral Image Classification. *IEEE Trans. Image Process.* **2018**, *27*, 1259–1270.

(37) Han, H.; Choi, S. Transfer Learning from Simulation to Experimental Data: NMR Chemical Shift Predictions. *J. Phys. Chem. Lett.* **2021**, *12*, 3662–3668.

(38) Ryder, A.; O'Connor, G.; Glynn, T. Quantitative Analysis of Cocaine in Solid Mixtures Using Raman Spectroscopy and Chemometric Methods. *J. Raman Spectrosc.* **2000**, *31*, 221–227.

(39) Li, S.; Gao, J.; Nyagilo, J. O.; Dave, D. P. Probabilistic Partial Least Square Regression: A Robust Model for Quantitative Analysis of Raman Spectroscopy Data. 2011 IEEE Int. Conf. Bioinform. Biomed. 2011; pp 526–531.

(40) Li, X.; Yang, T.; Li, S.; Wang, D.; Song, Y.; Zhang, S. Raman Spectroscopy Combined with Principal Component Analysis and k Nearest Neighbour Analysis for Non-Invasive Detection of Colon Cancer. *Laser Phys.* **2016**, *26*, 035702.

(41) Widjaja, E.; Zheng, W.; Huang, Z. Classification of Colonic Tissues Using Near-Infrared Raman Spectroscopy and Support Vector Machines. *Int. J. Oncol.* **2008**, *32*, 653–662.

(42) Madden, M. G.; Ryder, A. G. Machine Learning Methods for Quantitative Analysis of Raman Spectroscopy Data. Opto-Ireland 2002: Optics and Photonics Technologies and Applications. 2003; pp 1130–1139.

(43) Koch, G.; Zemel, R.; Salakhutdinov, R. Siamese Neural Networks for One-Shot Image Recognition. Proc. Representation Learning Workshop at ICML. 2015.

(44) Lafuente, B.; Downs, R. T.; Yang, H.; Stone, N. *The Power of Databases: The RRUFF Project*; 2015; pp 1–30.

(45) Virtanen, P. et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **2020**, *17*, 261–272.

(46) LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324.

(47) Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. 2017 IEEE International Conference on Computer Vision (ICCV). 2017.

(48) Alves, J. F.; Edwards, H. G. M.; Korsakov, A.; de Oliveira, L. F. C. Revisiting the Raman Spectra of Carbonate Minerals. *Minerals* **2023**, *13*.

(49) Reddy, B. J.; Frost, R. L.; Martens, W. N. Characterization of conichalcite by SEM, FTIR, Raman and electronic reflectance spectroscopy. *Mineralogical Magazine* **2005**,

(50) Ferraro, J. R., Nakamoto, K., Brown, C. W., Eds. *Introductory Raman Spectroscopy, 2nd ed.*; Academic Press, 2003; p iii.

(51) Liu, H.; Cong, X.; Lin, M.-L.; Tan, P.-H. The Intrinsic Temperature-Dependent Raman Spectra of Graphite in the Temperature Range from 4 K to 1000 K. *Carbon* **2019**, *152*, 231–239.

(52) Odena, A. Semi-Supervised Learning with Generative Adversarial Networks. 2016; arXiv preprint arXiv:1606.01583.

(53) Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved Techniques for Training GANs. Adv. Neural Inf. Process. Syst. 2016.

(54) Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. Proc. Int. Conf. Mach. Learn. 2020; pp 1597–1607.

(55) Dai, Z.; Yang, Z.; Yang, F.; Cohen, W. W.; Salakhutdinov, R. R. Good Semi-Supervised Learning That Requires a Bad GAN. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.

(56) Xiong, W.; Du, J.; Wang, W. Y.; Stoyanov, V. Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model. 2019; arXiv preprint arXiv:1912.09637.

(57) Corpolongo, A. et al. SHERLOC Raman Mineral Class Detections of the Mars 2020 Crater Floor Campaign. *J. Geophys. Res. Planets* **2023**, *128*, e2022JE007455.

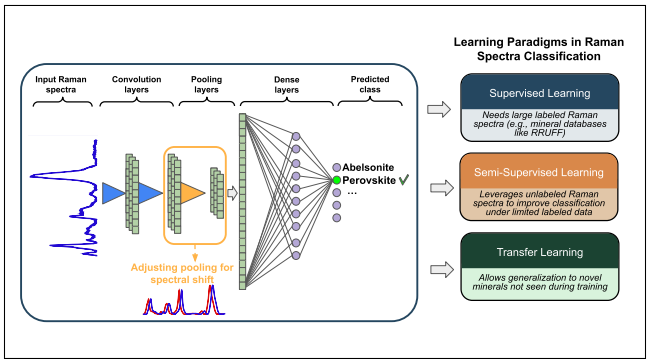
(58) Maurice, S. et al. The SuperCam Instrument Suite on the Mars 2020 Rover: Science Objectives and Mast-Unit Description. *Space Sci. Rev.* **2021**, *217*.

(59) Yanchilina, A.; Rodriguez, L. E.; Price, R.; Barge, L.; Sobron, P. Sensing Remote Realms of the Deep Ocean on Earth—and Beyond. *Eos* **2024**, *105*.

(60) Takahashi, T.; Takahagi, W.; Tasumi, E.; Makabe, A.; Taguchi, K.; Thornton, B.; Takai, K. In Situ Measurement of Liquid and Gas CO2 with High Purity at Deep-Sea Hydrothermal Vents in the Mariana Arc Using Raman Spectroscopy. *ACS Earth Space Chem.* **2023**, *7*, 2489–2497.

(61) Vitkova, A.; Vu, T. H.; Lambert, J. Extended Longevity Photoactivated Surface-Enhanced Raman Spectroscopy for the Detection of Biosignatures on Icy Worlds and Martian Polar Caps. *J. Raman Spectrosc.* **2024**,

TOC Graphic


Learning Paradigms in Raman Spectra Classification
Input Raman spectra
Convolution layers
Pooling layers
Dense layers
Predicted class
Abelsonite
Perovskite
...
Adjusting pooling for spectral shift
Supervised Learning
Needs large labeled Raman spectra (e.g., mineral databases like RRUFF)
Semi-Supervised Learning
Leverages unlabeled Raman spectra to improve classification under limited labeled data
Transfer Learning
Allows generalization to novel minerals not seen during training

Supporting Information: Reevaluating Convolutional Neural Networks for Spectral Analysis: A Focus on Raman Spectroscopy

Deniz Soysal,[†] Xabier García–Andrade,[†] Laura E. Rodriguez,[‡] Pablo Sobron,[¶] Laura M. Barge,[§] and Renaud Detry[*,‖,⊥]

[†]*KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium*

[‡]*Lunar & Planetary Institute, Universities Space Research Association, 3600 Bay Area Boulevard, Houston, TX 77058, USA*

[¶]*Impossible Sensing, 20 South Sarah Street, St. Louis, MO 63108, USA*

[§]*Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, La Cañada Flintridge, CA 91011, USA*

[‖]*KU Leuven, Dept. Electrical Engineering, Research Unit Processing Speech and Images (PSI), Kasteelpark Arenberg 10, 3001 Leuven, Belgium*

[⊥]*KU Leuven, Dept. Mechanical Engineering, Research Unit Robotics, Automation and Mechatronics (RAM), Celestijnenlaan 300, 3001 Leuven, Belgium*

E-mail: renaud.detry@kuleuven.be

This Supporting Information provides a high-level overview of Raman spectroscopy and machine learning methodologies, aiming to bridge the knowledge gap between spectroscopy experts and machine learning practitioners. It consists of 29 pages and 20 figures.

Raman Spectroscopy

Raman spectroscopy relies on the principle of **inelastic scattering**[1] of photons by a material. This scattering process is governed by the vibrational modes of molecules, which depend on both chemical bonding and structural symmetry. Consequently, a Raman spectrum serves as a **molecular fingerprint**, enabling the identification and structural analysis of chemical compounds.

Figure S1 illustrates the typical workflow of Raman spectroscopy. A laser illuminates the sample, causing the molecules to scatter light. The frequency shift between the incident and scattered photons corresponds to molecular vibrations and encodes information about the sample's chemical structure. The scattered light is collected by a lens and directed through a monochromator to produce a spectrum.

The resulting raw spectrum plots signal intensity as a function of Raman shift (in cm^{-1}). To enhance interpretability, this signal is typically preprocessed. Standard preprocessing steps include baseline correction, spectral truncation to focus on informative regions, and normalization. From the processed spectrum, scientists identify peak positions and intensities to determine the molecular composition of the sample.

Data-Driven Signal Interpretation

Machine learning (ML) provides effective tools for interpreting complex signals, including Raman spectra. A central goal in ML is to learn a function that maps inputs to outputs using a dataset of observed examples. In this context, the input is typically a Raman spectrum, and the output is a corresponding mineral class.

Formally, a classification dataset consists of M samples $\{(x_i, y_i)\}_{i=1}^{M}$, where $x_i \in \mathcal{X}$ represents an input instance (e.g., a spectrum), and $y_i \in \mathcal{Y}$ is the associated label. The input space

[1] In inelastic scattering, the incident photon interacts with a molecule, resulting in a gain or loss of energy. In contrast, elastic scattering conserves energy. Raman spectroscopy exploits inelastic scattering to probe molecular vibrations.

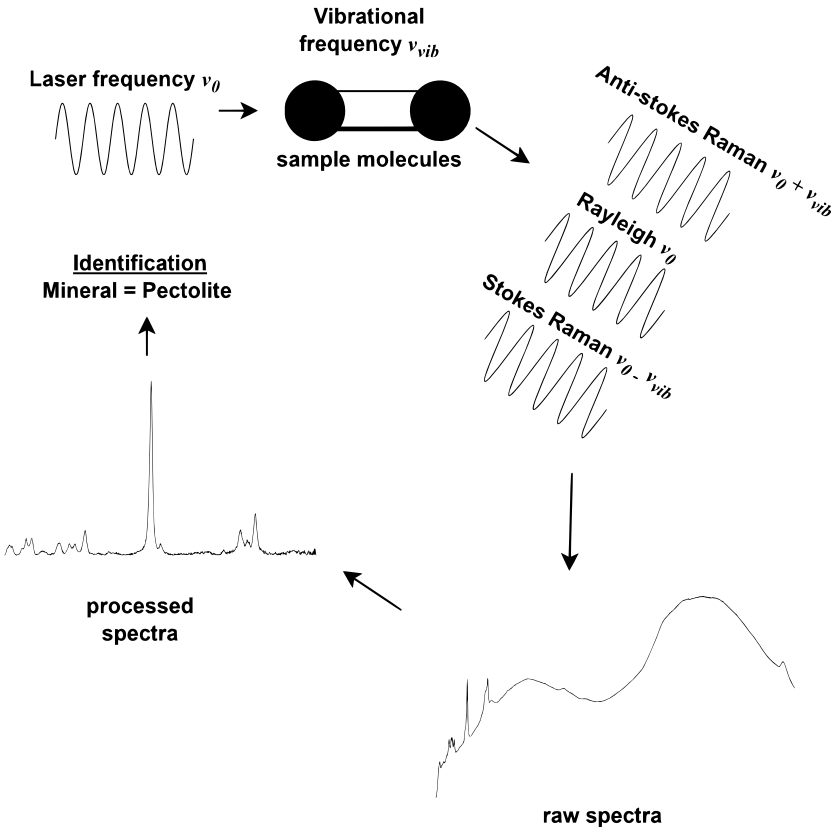


Figure S1: Schematic representation of Raman spectroscopy for mineral identification. A monochromatic laser source excites molecular vibrations within the sample, resulting in inelastic scattering. The spectrum consists of Rayleigh (elastic), Stokes, and anti-Stokes (inelastic) components, depending on the vibrational frequency ν_{vib}. The raw spectrum is then preprocessed to remove baseline drift and enhance resolution, enabling accurate identification of the mineral species—in this case, Pectolite.

$\mathcal{X}$ is often multidimensional. In Raman spectroscopy, x_i is usually a vector in $\mathbb{R}^S$, where S denotes the number of sampled Raman shift positions. Additional contextual features, such as the laser excitation wavelength, can be included, in which case $x_i \in \mathbb{R}^S \times \mathbb{R}^+$.

The label space $\mathcal{Y}$ consists of discrete values representing the possible classes, so that $y_i \in \{1, 2, \ldots, N\}$, where N is the number of mineral types. The goal is to approximate a mapping function $f_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by θ, that predicts the correct label y_i given an input x_i.

To train this model, a loss function $L(y_i, f(x_i))$ quantifies the error between the predicted and actual labels. A widely used loss for multi-class classification is the categorical cross-

entropy:

$$L(y_i, f(x_i)) = -\sum_{k=1}^{N} y_{ik} \log \hat{y}_{ik}, \tag{1}$$

where $\hat{y}_{ik} = f(x_i)$ is the predicted probability of class k, and y_{ik} is a one-hot encoded ground truth label. The model is trained by minimizing the total loss over the dataset:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{M} L(y_i, f_\theta(x_i)). \tag{2}$$

The discussion thus far assumes a supervised learning setting, where each spectrum x_i has an associated ground-truth label y_i. In contrast, unsupervised learning methods operate on unlabeled data and aim to uncover structure within the input space—for instance, clustering spectra that likely belong to the same mineral class. Techniques such as k-means clustering,[S1] principal component analysis (PCA),[S2,S3] and generative modeling[S4] are commonly applied.

Adopting ML for spectral classification arises from the intrinsic difficulty of writing explicit rules to differentiate mineral classes based solely on Raman features. Spectral variability, peak overlaps, and noise make manual rule-based approaches impractical. In contrast, ML models trained on labeled examples can automatically learn discriminative patterns in the data, without requiring hand-crafted rules or domain-specific heuristics.

Having introduced the fundamental concepts and terminology, the remainder of this section focuses on several representative ML methods applied in the classification of Raman spectra: Support Vector Machines (SVMs), k-Nearest Neighbors (KNN), and deep learning techniques such as Convolutional Neural Networks (CNNs).

Support Vector Machines (SVM)

Support Vector Machines (SVMs), introduced by Vapnik *et al.*,[S5] are methods used for classification and function estimation tasks. Their core principle is *margin maximization*: identifying a hyperplane that separates data from different classes with the largest possible

margin. This margin corresponds to the distance between the hyperplane and the closest samples from each class, known as *support vectors*. These critical data points determine the orientation and position of the decision boundary.

Linear Case Consider a binary classification dataset $\{(x_k, y_k)\}_{k=1}^{N}$, where $x_k \in \mathbb{R}^S$ are input features and $y_k \in \{-1, +1\}$ are class labels. A linear SVM seeks a weight vector W and bias b such that the decision function

$$\hat{y} = \text{sign}(W^\top x + b) \tag{3}$$

correctly classifies the training data. To ensure correct classification, the model must satisfy:

$$\begin{aligned} W^\top x_k + b &\geq 1 \quad \text{if } y_k = +1, \\ W^\top x_k + b &\leq -1 \quad \text{if } y_k = -1. \end{aligned} \tag{4}$$

These constraints can be unified as:

$$y_k(W^\top x_k + b) \geq 1 \quad \forall k. \tag{5}$$

The optimal hyperplane is obtained by solving the following convex optimization problem:

$$\begin{aligned} \min_{W,b} \quad & \frac{1}{2}\|W\|^2 \\ \text{subject to} \quad & y_k(W^\top x_k + b) \geq 1, \quad \forall k. \end{aligned} \tag{6}$$

This formulation maximizes the margin $2/\|W\|$ between the two classes. However, real-world data often contain noise or overlapping classes, which make the data not perfectly separable. To accommodate misclassifications, slack variables $\xi_k \geq 0$ are introduced, yielding the soft-margin SVM:

$$\begin{aligned} \min_{W,b,\xi} \quad & \frac{1}{2}\|W\|^2 + C\sum_{k=1}^{N}\xi_k \\ \text{subject to} \quad & y_k(W^\top x_k + b) \geq 1 - \xi_k, \quad \xi_k \geq 0, \end{aligned} \tag{7}$$

where $C > 0$ is a regularization parameter controlling the trade-off between maximizing the margin and minimizing classification errors.

Nonlinear Case When the data are not linearly separable, SVMs can still be applied by transforming the input into a higher-dimensional space using a mapping $\Phi(x)$. In this transformed space, a linear hyperplane may be sufficient for separation:

$$\hat{y} = \text{sign}(W^\top \Phi(x) + b). \tag{8}$$

Direct computation in high-dimensional feature spaces can be computationally expensive. To address this, the *kernel trick* is used: inner products $\Phi(x)^\top \Phi(y)$ are replaced by a kernel function $K(x, y)$, allowing the SVM to operate implicitly in the high-dimensional space:

$$K(x, y) = \Phi(x)^\top \Phi(y). \tag{9}$$

Popular kernel functions include the polynomial kernel, the radial basis function (RBF) kernel, and the sigmoid kernel. The choice of kernel significantly impacts the model's expressiveness and generalization.

Key Hyperparameters The performance of an SVM depends on proper tuning of the following hyperparameters:

- **Kernel function** $K(x, y)$: Defines the nature of the non-linear transformation applied to the data. The choice of kernel affects the flexibility of the decision boundary
- **Regularization parameter** C: Balances margin maximization with classification error. A smaller C allows for a wider margin but may increase misclassifications (un-

derfitting), while a larger C aims to classify all training examples correctly but may lead to a smaller margin and overfitting.

- **Kernel coefficient** γ: Defines the distance of influence of a data point. For a radial basis function kernel, it is inversely proportional to the variance of the Gaussian distribution. For low values of γ, further data points are considered as belonging to the same class. For high values of γ, the data points need to be closer to be considered as belonging to the same class.

Practical Considerations Several practical aspects must be addressed when applying SVMs:

- **Feature scaling** is essential to prevent features with larger magnitudes from dominating the optimization. Standardization or normalization is recommended.
- **Computational cost** can be significant for large datasets, as SVMs solve a quadratic programming problem. Efficient solvers and approximations may be needed for scalability.
- **Memory usage** can be high, especially when many support vectors are retained. This also affects prediction speed.
- **Model selection** requires careful cross-validation to choose appropriate values for C, γ, and the kernel type.
- **Class imbalance** can bias the model toward majority classes. Techniques such as class weighting or resampling are often necessary.
- **Outlier sensitivity** can degrade performance, as outliers may become support vectors. Preprocessing to detect and remove such points improves robustness.

k-Nearest Neighbors (KNN)

The k-nearest neighbors (KNN) algorithm, introduced by Cover *et al.*,[S6] is a non-parametric method used for classification and regression. In classification tasks, KNN assigns a label to an unlabeled data point $\hat{x}$ based on the majority class among its k closest neighbors in the training set. Unlike model-based methods such as SVM, KNN is an instance-based approach. It does not build an explicit model during training but instead retains the entire dataset and performs classification at inference time.

To classify a new point $\hat{x}$, the algorithm computes its distance to each sample x_i in the training set $X = \{x_1, x_2, \ldots, x_M\}$ using a specified distance metric $D(x_i, \hat{x})$. Common choices include the Euclidean, Manhattan, and Mahalanobis distances. After ranking the distances, the k closest samples are selected, and the most frequent class among them is assigned to $\hat{x}$.

Key Parameters Two primary parameters influence KNN performance:

- **Distance metric**: Determines how similarity is measured between samples. The choice of metric affects how neighborhoods are formed in the feature space.
- **Number of neighbors** k: Controls the smoothness of the decision boundary. A small k (e.g., $k = 1$) makes the algorithm highly sensitive to noise (overfitting), while a larger k smooths out decision boundaries but may reduce sensitivity to local patterns (underfitting).

Practical Considerations Several factors should be considered when applying KNN in practice:

- **Feature scaling** is essential, as KNN relies on distance computations. Unscaled features can disproportionately influence results. Standardization or normalization is typically required.

- **Prediction cost** is high because the algorithm must compute distances to all training points at inference time. Efficient data structures (e.g., KD-trees or ball trees) can partially mitigate this cost.

- **Memory usage** grows linearly with dataset size since all training data must be stored. This can become a limiting factor for large datasets.

- **Curse of dimensionality** affects KNN performance in high-dimensional spaces, where distances lose discriminative power. Dimensionality reduction (e.g., PCA) is often necessary.

- **Sensitivity to noise and irrelevant features** can degrade classification accuracy. Feature selection or extraction helps improve robustness.

- **Class imbalance** may cause KNN to favor the majority class. Weighting neighbors inversely by distance or adjusting class priors can mitigate this issue.

- **Model selection** involves choosing the optimal value of k via cross-validation, similar to hyperparameter tuning in SVMs.

Deep Learning

After discussing traditional machine learning approaches such as Support Vector Machines and k-Nearest Neighbors, we now turn to deep learning, with a particular focus on neural networks.

While conventional algorithms can be effective for certain tasks, they often rely on hand-crafted features and may struggle to capture the complex, high-dimensional patterns inherent in Raman spectra. These spectra frequently exhibit overlapping, subtle, and nonlinear features resulting from diverse molecular vibrations. Deep learning models, particularly neural networks with multiple layers, have demonstrated strong capabilities in modeling such complexity. By learning hierarchical representations directly from raw data, these models can

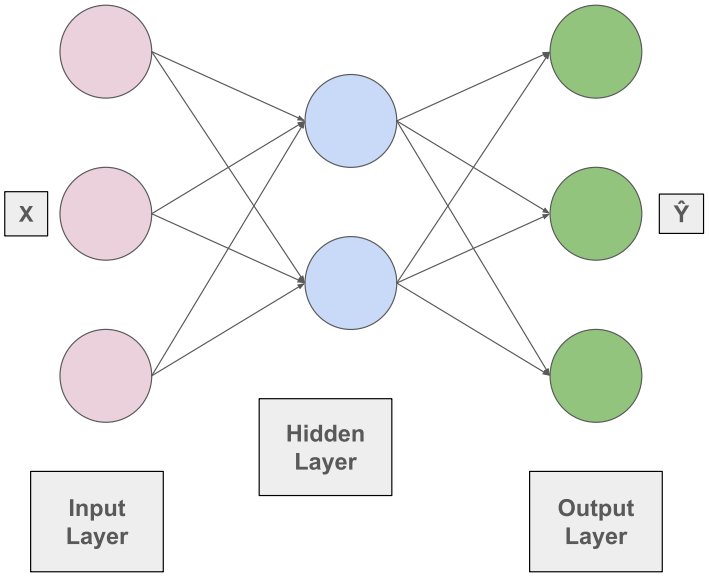


Figure S2: Schematic representation of a multilayer perceptron (MLP).

automatically extract discriminative features, making them especially suitable for Raman spectral classification.

Deep learning is founded on artificial neural networks—computational architectures inspired by biological neural systems. A neural network learns to model relationships between inputs and outputs by adjusting its internal parameters during training.

Neural networks consist of layers of interconnected computational units called *neurons*, as shown in Figure S2. These layers include an input layer, one or more hidden layers, and an output layer. In this discussion, we focus on the multilayer perceptron (MLP) with a single hidden layer, although deeper networks are commonly used in practice.

Each neuron is associated with two parameters: weights (θ) and a bias term (b). When an input vector X is fed into the network, each neuron performs a linear transformation:

$$Z = \theta \cdot X + b, \tag{10}$$

followed by a non-linear activation function $g(\cdot)$ to introduce non-linearity into the model:

$$A = g(Z) = g(\theta \cdot X + b), \tag{11}$$

where A denotes the neuron's activation. These computations are applied layer by layer in what is known as the *feedforward* step, culminating in a network output $\hat{y}$.

Learning occurs during the *backward* step through a process called *backpropagation*, which uses the chain rule to compute the gradients of the loss function with respect to each parameter in the network. These gradients are then used to update the parameters via gradient descent. For instance, the partial derivatives required during backpropagation are given by:

$$\frac{\partial L}{\partial \theta^{[L]}} = \frac{1}{N} \frac{\partial L}{\partial Z^{[L]}} A^{[L-1]\top}, \tag{12}$$

$$\frac{\partial L}{\partial b^{[L]}} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L}{\partial Z^{[L](i)}}, \tag{13}$$

$$\frac{\partial L}{\partial A^{[L-1]}} = \left(\theta^{[L]}\right)^{\top} \frac{\partial L}{\partial Z^{[L]}}, \tag{14}$$

where L is the loss function, $\theta^{[L]}$ the weights in layer L, $Z^{[L]}$ the pre-activation output, $A^{[L-1]}$ the activation from the previous layer, and N the number of training samples.

This training process is repeated iteratively over multiple epochs until convergence. The network progressively adjusts its parameters to minimize the discrepancy between predicted and true outputs.

Although the concept of MLPs dates back several decades, the resurgence of deep learning has been driven by the availability of large datasets and advances in computing hardware. These developments have enabled the training of *deep* neural networks, which contain many layers and millions of parameters, dramatically increasing their representational power.

0.0.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models originally developed for image processing and now widely used across various domains. From LeCun's early work on handwritten digit recognition in 1998[S7] to breakthroughs in large-scale image classification tasks such as ImageNet,[S8] CNNs have played a central role in advancing modern artificial intelligence.

CNNs extract hierarchical features from raw input data by convolving learnable filters (also called kernels) across the input space. Unlike traditional convolution operations with fixed filters, CNNs learn the filter weights during training via backpropagation, much like the weights in Multilayer Perceptrons (MLPs). This capacity to learn informative features directly from raw data makes CNNs particularly effective for modeling high-dimensional input signals.

A fundamental advantage of CNNs over fully connected networks is *weight sharing*. A single filter is applied repeatedly across different positions in the input, allowing the network to detect spatially recurring patterns while reducing the number of parameters. This property also confers *translational invariance*, meaning the network can recognize a feature regardless of its position in the input. These architectural advantages make CNNs well suited for tasks involving spatially structured inputs, such as images or sequential data.

A typical CNN architecture consists of three main types of layers: convolutional layers, pooling layers, and fully connected layers. Although real-world models use multiple filters, we will illustrate the core principles using a single-filter example.

Convolutional Layers Convolutional layers form the foundation of CNNs. Each layer applies a set of filters to the input by sliding them over the signal and computing element-wise multiplications followed by a summation. Figure S3 illustrates a 2D convolution example, where a 3×3 kernel is applied to a single-channel input. The output at each location is computed as the dot product between the filter and the corresponding local region of the

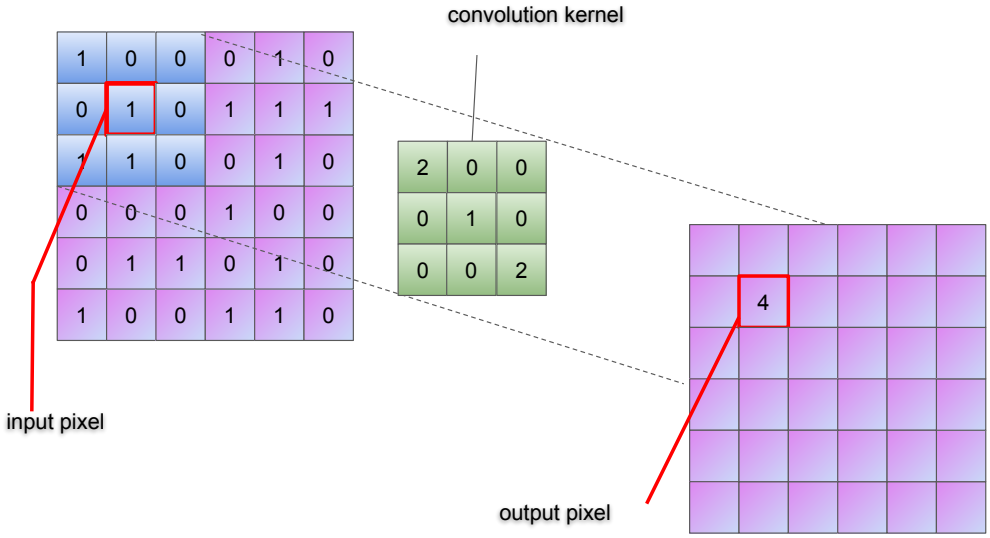


Figure S3: Example of a 2D convolution operation using a 3×3 kernel on a single-channel image.

input.

For instance, consider a 3×3 kernel with values arranged along the diagonal as $\{2, 1, 2\}$. If the input region matches the kernel pattern (e.g., values $\{1, 1, 0\}$), the resulting output would be high, e.g., $1 \times 2 + 1 \times 1 + 0 \times 2 = 3$. More formally, the 1D convolution operation can be written as:

$$S(t) = (x * w)(t) = \sum_{n} x(n)w(t - n), \tag{15}$$

where $x(n)$ denotes the input signal and $w(n)$ the convolution kernel. Convolution is a linear operation, so to model non-linear patterns, a non-linear activation function is applied to the output. Common activation functions include the Rectified Linear Unit (ReLU), sigmoid, and hyperbolic tangent.

Relevance to Raman Spectroscopy Although Raman spectra are one-dimensional signals, we include this explanation of 2D convolution to clarify CNN principles and highlight the challenges of applying CNNs to spectral data. In 2D image tasks, CNNs leverage in-

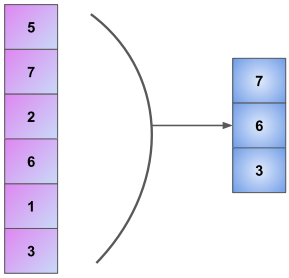


Figure S4: Illustration of a 1D max-pooling operation with a window size of 2.

ductive biases such as *local connectivity* and *spatial invariance*—both advantageous when features can appear anywhere in the image.

However, in Raman spectroscopy, this invariance is less desirable. Spectral peaks are localized and occur at specific Raman shifts that are critical for identifying molecular structures. If a CNN applies spatial invariance to Raman spectra, it may treat peaks at different positions as equivalent, potentially discarding vital information about molecular identity.

Therefore, adapting CNN architectures for Raman analysis requires careful modification of these biases. In particular, preserving the absolute positions of peaks is essential. Architectures for spectral data should use one-dimensional convolutions with reduced pooling to retain resolution along the spectral axis. These changes ensure that CNNs can effectively model the precise spectral patterns needed for accurate classification.

Pooling Layers Pooling layers reduce the spatial dimensionality of feature maps by summarizing local responses, thereby improving computational efficiency and helping mitigate overfitting. The most widely used pooling operation is *max pooling*, which retains the maximum value within a specified window of size m^2. This process preserves the most prominent features while discarding less significant variations. Figure S4 shows a one-dimensional (1D) max-pooling operation with $m = 2$.

[2]An alternative approach is average pooling, which computes the mean value within the same window.

While pooling is effective in image analysis—where features can appear in various positions and translation invariance is desirable—its application to Raman spectral data must be handled with care. In Raman spectroscopy, the absolute positions of spectral peaks along the Raman shift axis are essential for accurate molecular identification. Excessive pooling may blur or eliminate these peaks, leading to diminished sensitivity to fine spectral features and possible misclassification. Therefore, architectures designed for Raman data should use pooling layers judiciously, with window sizes selected to preserve the positional integrity of key spectral information.

Fully Connected Layers A typical CNN architecture consists of a sequence of convolutional and pooling layers, followed by one or more fully connected layers. In convolutional layers, each neuron is locally connected—that is, it receives input from only a small, spatially contiguous region of the preceding layer. For example, as shown in Figure S3, the highlighted output neuron is influenced by a 3×3 region (nine neurons) from the previous feature map.

In contrast, fully connected layers establish connections between every neuron in one layer and all neurons in the preceding layer. This dense connectivity enables the network to integrate the local patterns extracted by the convolutional layers and to model complex, global relationships across the entire input. When paired with nonlinear activation functions, fully connected layers allow the model to learn intricate combinations of features, which is particularly useful for classification tasks.

Figure S5 illustrates a representative CNN architecture. The input image is shown in blue, followed by convolutional feature maps in yellow, a pooled feature map in orange, and fully connected layers in purple and green. These final layers synthesize the extracted features and map them to the output space, completing the decision-making process of the network.

1D CNN for Raman Spectra While the CNN architecture shown in Figure S5 is designed for two-dimensional (2D) inputs such as images, Raman spectra are fundamentally

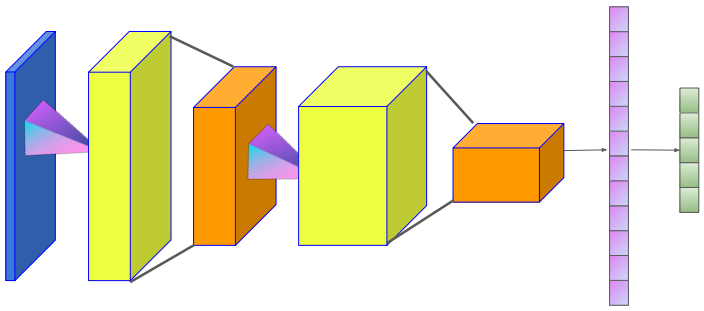


Figure S5: Illustrative example of a typical CNN architecture.

one-dimensional (1D) signals, with intensity values indexed by Raman shift. To process these spectral data effectively, we employ a one-dimensional convolutional architecture tailored to the structure of Raman signals.

Figure S6 illustrates a simplified example of a 1D convolution applied to a binary input vector, where a value of 1 indicates the presence of a peak and 0 indicates its absence. The convolution kernel in this example is defined as $[-2, 4, -2]$, designed to activate strongly when centered on a peak. The convolution at a given position t is computed as:

$$S(t) = \sum_{n} x(n) \cdot w(t - n), \tag{16}$$

where $x(n)$ denotes the input signal, $w(n)$ the kernel weights, and $S(t)$ the output signal. For example, if the peak is centered at position t, the kernel response evaluates to $S(t) = (-2 \times 0) + (4 \times 1) + (-2 \times 0) = 4$.

Crucially, in a CNN, the kernel weights are not fixed but are learned during training via backpropagation. This allows the network to discover discriminative patterns in the data—such as peak positions, widths, and intensities—directly from raw spectra without requiring manual feature engineering.

While pooling layers can provide limited translation invariance, their use in spectral data must be approached with caution. In Raman spectroscopy, the absolute positions of spectral

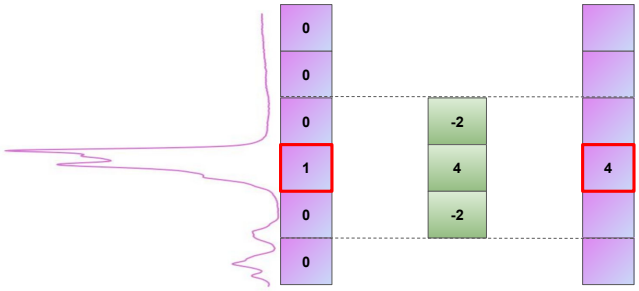


Figure S6: Illustration of a 1D convolution applied to a Raman spectrum using a kernel that responds selectively to peaks.

peaks carry significant chemical information. Excessive pooling may blur these features and degrade classification performance. To mitigate this, one can limit the depth or stride of pooling layers, or omit them entirely in favor of preserving high resolution along the spectral axis. This design choice allows the model to remain sensitive to meaningful spectral shifts while maintaining robustness against minor experimental noise.

Semi-Supervised Learning

In many real-world scenarios, labeled data are scarce while unlabeled data are abundant. Semi-supervised learning provides a framework to exploit this imbalance by using unlabeled data to improve model performance when labels are limited. The central assumption is that the posterior class distribution $p(\mathcal{C}|\mathbf{X})$ can be inferred more effectively when incorporating knowledge of the marginal input distribution $p(\mathbf{X})$.[S9] In other words, understanding the structure of the input space can help refine the classification boundary, even without direct access to labels.

Figure S7 shows a conceptual example of binary classification. The colored points represent labeled data, while the gray points represent unlabeled samples. In the absence of unlabeled data (left panel), the decision boundary is overly simplistic. Incorporating the unlabeled data (right panel) enables the model to learn a boundary that better reflects the

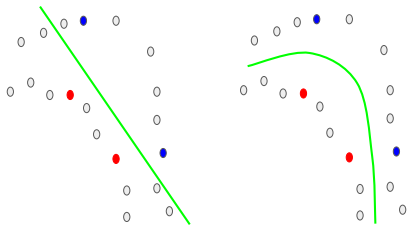


Figure S7: Illustrative example of semi-supervised learning. Left: decision boundary based only on labeled data. Right: boundary refined using unlabeled data.

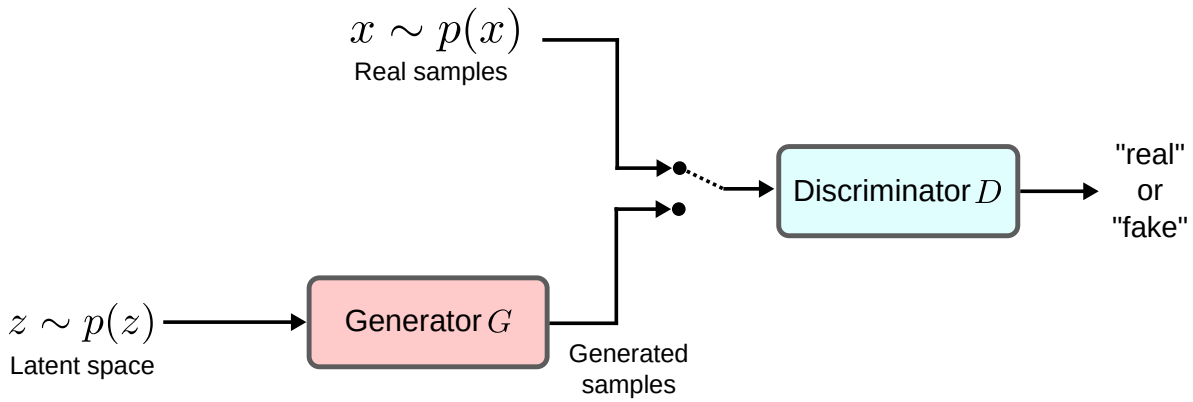


Figure S8: Architecture of a Generative Adversarial Network (GAN). Adapted from.[S10]

true data distribution. This refinement relies on the *smoothness assumption*, which posits that nearby points in the input space are likely to share the same label.

0.0.2 Semi-Supervised Generative Adversarial Networks

Generative Adversarial Networks (GANs) can be extended to semi-supervised learning, resulting in Semi-Supervised GANs (SGANs), which leverage both labeled and unlabeled data to improve classification performance.

Generative Adversarial Networks As illustrated in Figure S8, a standard GAN consists of two neural networks trained in opposition: a Generator (G) and a Discriminator (D). The Generator learns to map random noise vectors $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$ to synthetic samples that resemble real data drawn from a distribution $p(\mathbf{X})$. The Discriminator receives inputs that are either real data or generated data from the Generator, and outputs the probability that a sample

is real.

The objective of G is to generate data that D cannot distinguish from real samples, while D aims to correctly identify whether an input is real or generated. Formally, the GAN optimization problem is expressed as:[S11]

$$\begin{aligned}\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] \\ + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))],\end{aligned} \tag{17}$$

where $D(\boldsymbol{x})$ is the predicted probability that $\boldsymbol{x}$ is a real sample, and $G(\boldsymbol{z})$ is the output of the Generator. As training progresses, G learns to approximate the true data distribution $p(\mathbf{X})$, enabling the generation of realistic synthetic samples. GANs are widely used in applications such as image synthesis and data augmentation, where the trained Generator produces new data samples, and the Discriminator is typically used only during training.

Semi-Supervised GANs To incorporate unlabeled data, the SGAN framework modifies the role of the Discriminator. Instead of performing binary classification (real vs. fake), D is trained to classify inputs into $K+1$ categories: the original K real classes and an additional class representing synthetic data generated by G.[S12]

The Discriminator receives three types of input: labeled real samples, unlabeled real samples, and generated (synthetic) samples. Its objectives are as follows:

1. Classify labeled real samples correctly into one of the K real classes.

2. Assign unlabeled real samples to one of the K real classes, acknowledging their authenticity.

3. Assign generated samples to the $(K+1)^{\text{th}}$ class, identifying them as synthetic.

This design enables D to simultaneously act as a classifier and a detector of synthetic data, using adversarial training to improve generalization.

The SGAN objective function combines a supervised loss term with an unsupervised adversarial loss:

$$
\begin{aligned}
L &= L_{\text{supervised}} + L_{\text{unsupervised}}, \\
L_{\text{supervised}} &= -\mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}} \log p_{\text{model}}(y \mid \boldsymbol{x}, y \leq K), \\
L_{\text{unsupervised}} &= -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log[1 - p_{\text{model}}(y = K+1 \mid \boldsymbol{x})] \\
&\quad - \mathbb{E}_{\boldsymbol{x}\sim G} \log[p_{\text{model}}(y = K+1 \mid \boldsymbol{x})],
\end{aligned}
\tag{18}
$$

Here, $L_{\text{supervised}}$ encourages accurate classification of labeled data, while $L_{\text{unsupervised}}$ penalizes the Discriminator for misclassifying real samples as synthetic and rewards it for correctly identifying synthetic samples. This formulation enables the model to benefit from unlabeled data and adversarial feedback, enhancing its capacity to generalize from limited supervision.

0.0.3 Contrastive Learning

In the context of unsupervised representation learning for Raman spectra, contrastive learning has emerged as a promising approach. Although initially introduced to improve few-shot learning in supervised settings, contrastive methods have since gained popularity in self-supervised learning frameworks, where label information is unavailable.[S13]

In this work, we adopt the SimCLR framework introduced by Chen *et al.*,[S14] which learns useful representations by maximizing agreement between different augmented views of the same sample. Specifically, two stochastic augmentations, $T(\cdot)$ and $T'(\cdot)$, are independently applied to each input sample, producing a positive pair. These augmented inputs are passed through an encoder network $f(\cdot)$ to extract high-level features, followed by a projection head $g(\cdot)$ that maps the encoded representations into a latent space. The similarity between representations is measured using cosine similarity:

$$\text{sim}(\mathbf{z}, \mathbf{z}') = \frac{\mathbf{z}^\top \mathbf{z}'}{\|\mathbf{z}\| \cdot \|\mathbf{z}'\|}, \tag{19}$$

where $\mathbf{z}$ and $\mathbf{z}'$ are the projected representations of the augmented inputs, and $\|\cdot\|$ denotes the Euclidean norm.

To train the network, a contrastive loss is applied that encourages the representations of positive pairs to be similar, while pushing apart those of negative pairs. For a positive pair (i, j) within a batch of $2N$ augmented examples, the contrastive loss is defined as:

$$L_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \tag{20}$$

where τ is a temperature parameter that controls the concentration of the distribution, and $\mathbb{1}_{[k \neq i]}$ is an indicator function equal to 1 when $k \neq i$ and 0 otherwise. Each positive pair is formed by two augmentations of the same original input, while the remaining $2(N-1)$ examples in the batch serve as negative samples.

For Raman spectroscopy, the effectiveness of contrastive learning strongly depends on appropriate data augmentations that preserve the underlying chemical identity of the sample. Useful augmentation strategies include:

- **Additive noise:** Mimicking measurement noise to increase robustness.
- **Intensity scaling:** Varying signal amplitude to reflect experimental variability.
- **Peak shifting:** Slightly translating spectral peaks to simulate calibration errors while maintaining chemical interpretability.

By leveraging these transformations, contrastive learning enables the model to discover latent structure in the spectral data without requiring labels, laying the groundwork for downstream tasks such as classification or clustering.

Training and Validation Loss Curves

To assess convergence and over-/under-fitting, we plot the training and validation loss curves for our MLP variants and the 1-D CNN on both RRUFF-raw and RRUFF-clean datasets.

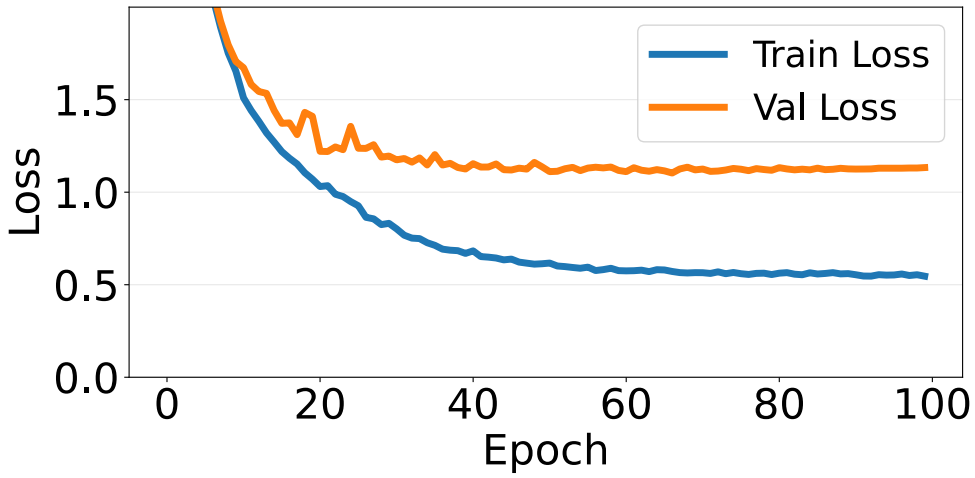


Figure S9: Training and validation categorical cross-entropy loss for **MLP-small** on the **RRUFF-raw** dataset. Persistent high validation loss indicates underfitting.

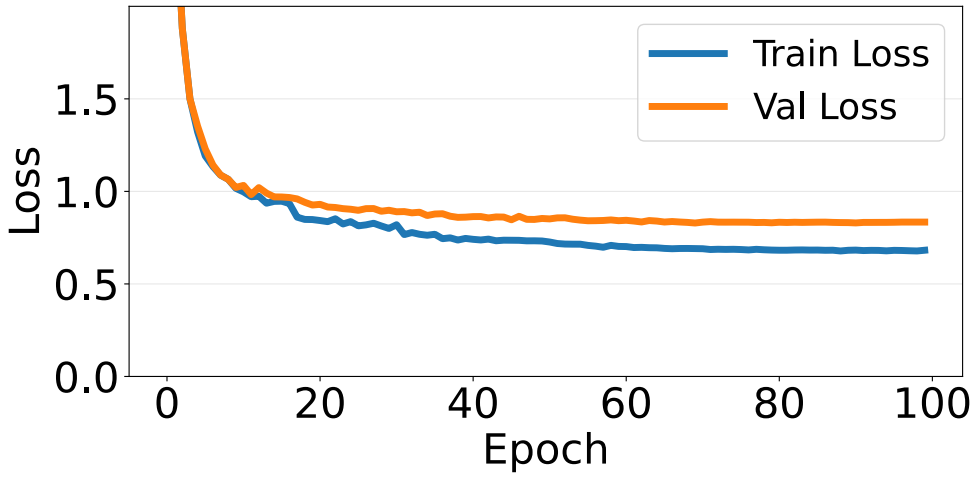


Figure S10: Training and validation categorical cross-entropy loss for **MLP-small** on the **RRUFF-clean** dataset.

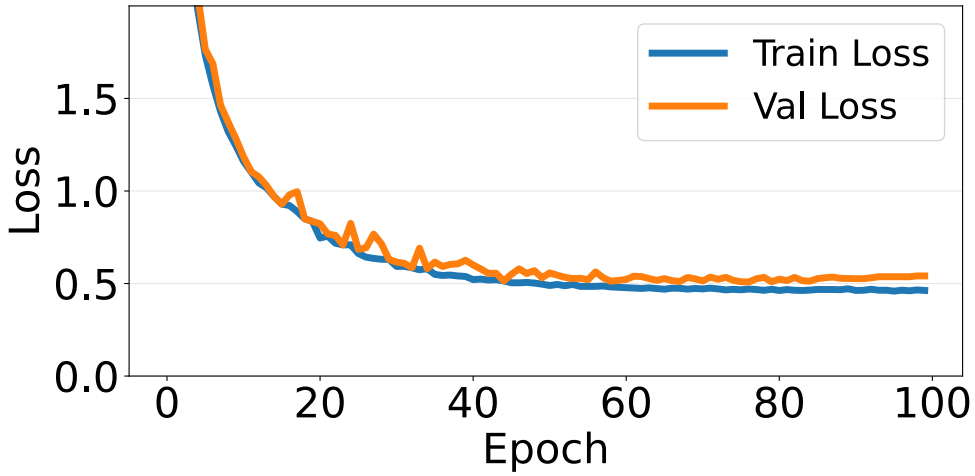


Figure S11: Training and validation categorical cross-entropy loss for **MLP-mid** on the **RRUFF-raw** dataset.

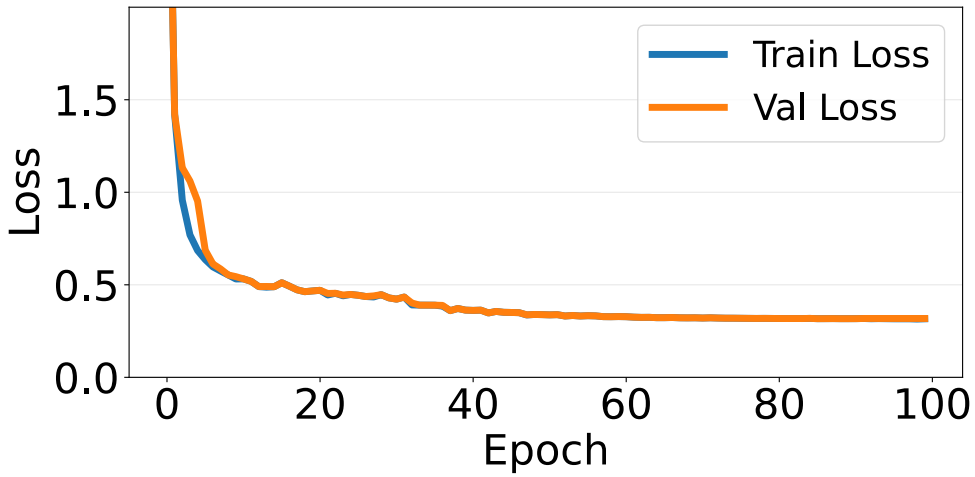


Figure S12: Training and validation categorical cross-entropy loss for **MLP-mid** on the **RRUFF-clean** dataset.

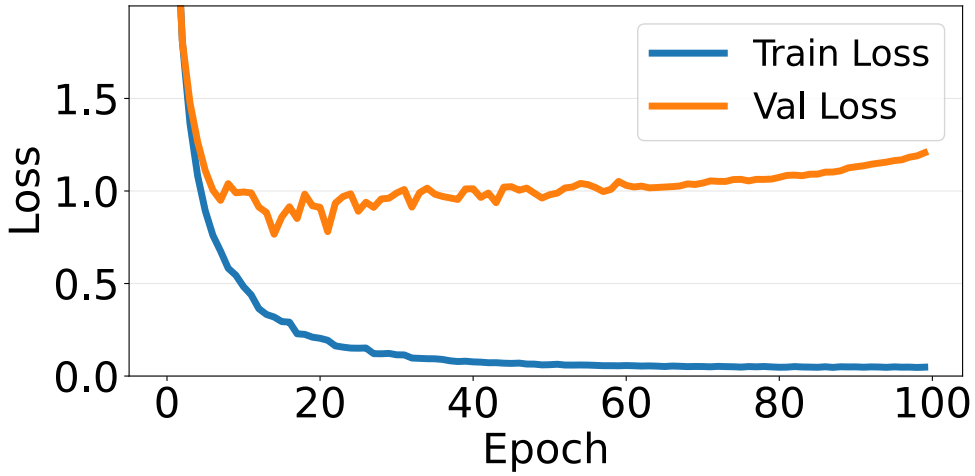


Figure S13: Training and validation categorical cross-entropy loss for **MLP-large** on the **RRUFF-raw** dataset. A widening gap indicates overfitting.

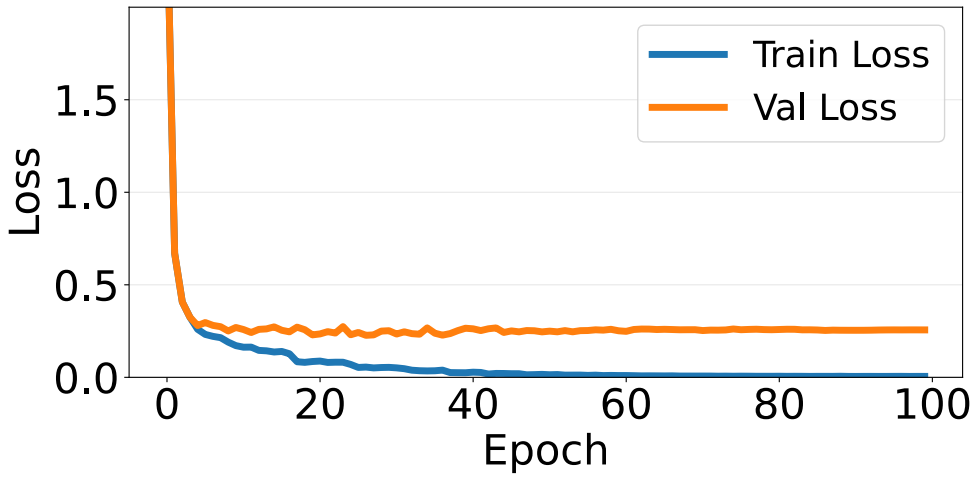


Figure S14: Training and validation categorical cross-entropy loss for **MLP-large** on the **RRUFF-clean** dataset.

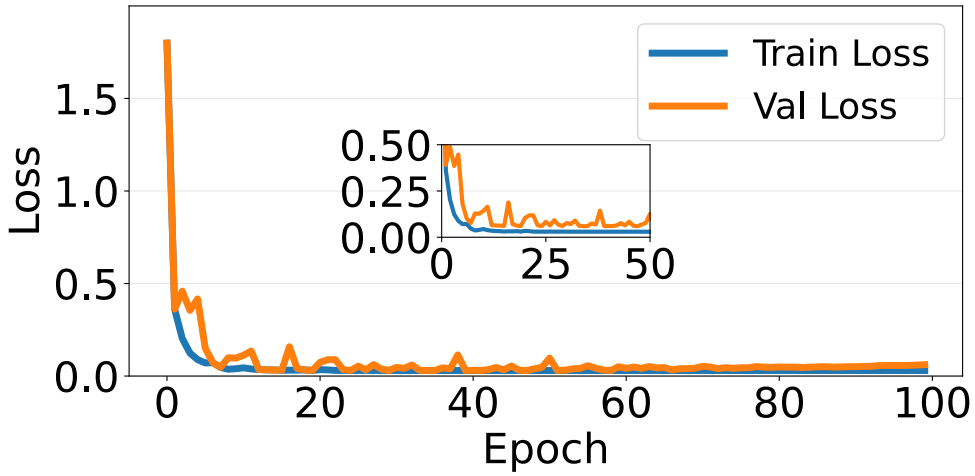


Figure S15: Training and validation categorical cross-entropy loss for **CNNs** on the **RRUFF-raw** dataset. Fast convergence and small generalization gap highlight strong fit.

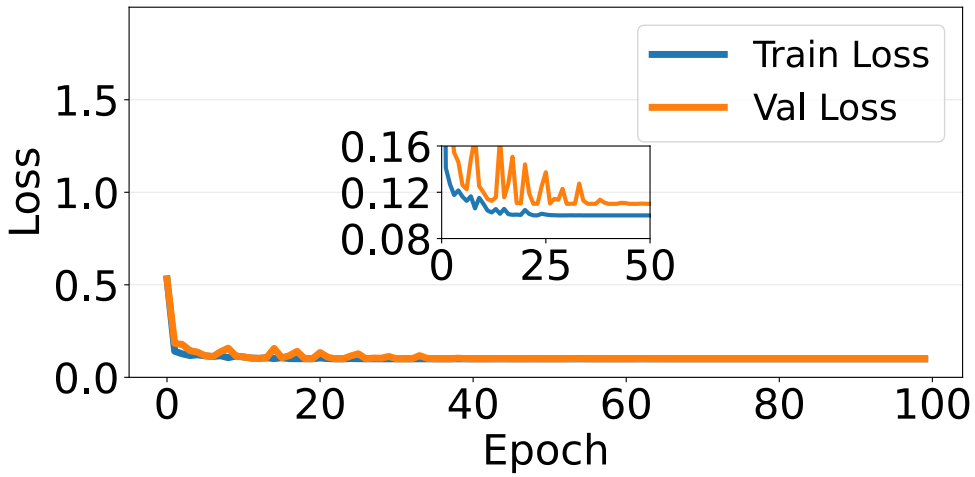


Figure S16: Training and validation categorical cross-entropy loss for **CNNs** on the **RRUFF-clean** dataset.

Transfer Learning Fine-Tuning Curves

Here we add the training curves for the transfer learning experiments.

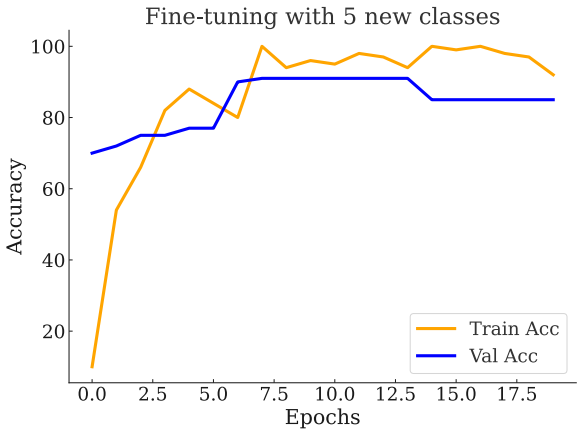


Figure S17: Fine-tuning training and validation Top-1 accuracy for $c = 5$ new classes.

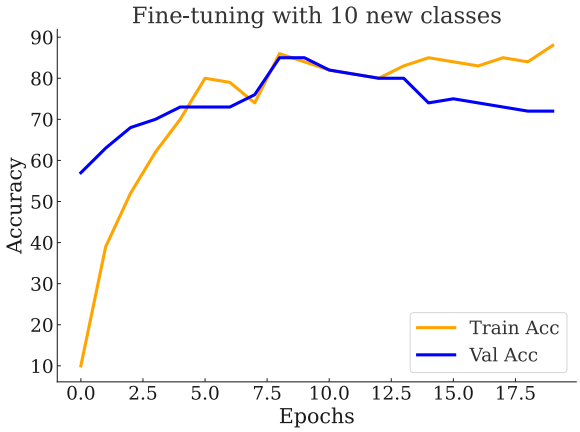


Figure S18: Fine-tuning training and validation Top-1 accuracy for $c = 10$ new classes.

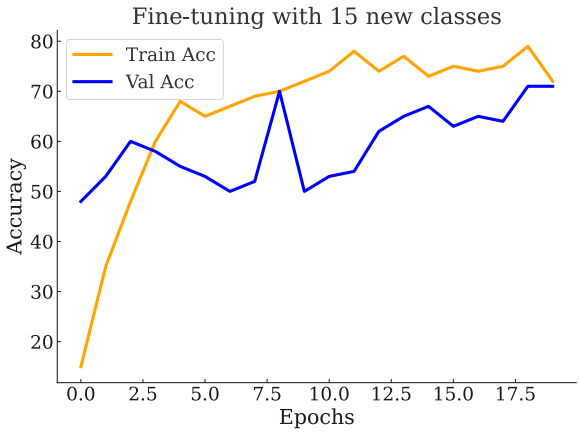


Figure S19: Fine-tuning training and validation Top-1 accuracy for $c = 15$ new classes.

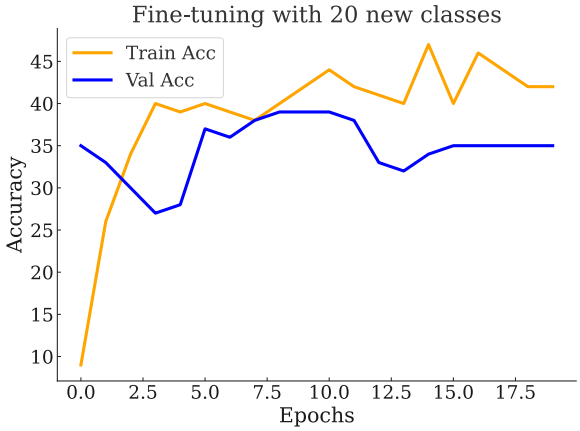


Figure S20: Fine-tuning training and validation Top-1 accuracy for $c = 20$ new classes.

References

(S1) Ball, G. H.; Hall, D. J. *ISODATA, a Novel Method of Data Analysis and Pattern Classification*; 1965.

(S2) Pearson, K. On Lines and Planes of Closest Fit to Systems of Points in Space. *Lond. Edinb. Dubl. Philos. Mag. J. Sci.* **1901**, *2*, 559–572.

(S3) Hotelling, H. Analysis of a Complex of Statistical Variables into Principal Components. *J. Educ. Psychol.* **1933**, *24*, 417–441.

(S4) Jebara, T. *Machine Learning: Discriminative and Generative*; Springer Science & Business Media, 2012; Vol. 755.

(S5) Vapnik, V. N. *The Nature of Statistical Learning Theory*; Springer-Verlag, 1995.

(S6) Cover, T.; Hart, P. Nearest Neighbor Pattern Classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27.

(S7) LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324.

(S8) Alom, M. Z.; Taha, T.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.; Hasan, M.; Essen, B.; Awwal, A.; Asari, V. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics* **2019**, *8*, 292.

(S9) van Engelen, J. E.; Hoos, H. H. A Survey on Semi-Supervised Learning. *Mach. Learn.* **2020**, *109*, 373–440.

(S10) Ouali, Y.; Hudelot, C.; Tami, M. An Overview of Deep Semi-Supervised Learning. 2020; arXiv preprint arXiv:2006.05278.

(S11) Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. Adv. Neural Inf. Process. Syst. 2014.

(S12) Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved Techniques for Training GANs. Adv. Neural Inf. Process. Syst. 2016.

(S13) Tian, Y.; Sun, C.; Poole, B.; Krishnan, D.; Schmid, C.; Isola, P. What Makes for Good Views for Contrastive Learning. *CoRR* **2020**, *abs/2005.10243*.

(S14) Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. Proc. Int. Conf. Mach. Learn. 2020; pp 1597–1607.