ARTICLE OPEN

Check for updates

# Validating neural networks for spectroscopic classification on a universal synthetic dataset

Jan Schuetzke[1], Nathan J. Szymanski[2,3] and Markus Reischl[1]✉

To aid the development of machine learning models for automated spectroscopic data classification, we created a universal synthetic dataset for the validation of their performance. The dataset mimics the characteristic appearance of experimental measurements from techniques such as X-ray diffraction, nuclear magnetic resonance, and Raman spectroscopy among others. We applied eight neural network architectures to classify artificial spectra, evaluating their ability to handle common experimental artifacts. While all models achieved over 98% accuracy on the synthetic dataset, misclassifications occurred when spectra had overlapping peaks or intensities. We found that non-linear activation functions, specifically ReLU in the fully-connected layers, were crucial for distinguishing between these classes, while adding more sophisticated components, such as residual blocks or normalization layers, provided no performance benefit. Based on these findings, we summarize key design principles for neural networks in spectroscopic data classification and publicly share all scripts used in this study.

*npj Computational Materials* (2023)9:100 ; https://doi.org/10.1038/s41524-023-01055-y

## INTRODUCTION

Spectroscopic techniques such as X-ray diffraction (XRD), Nuclear Magnetic Resonance (NMR), and Raman scattering are fundamental tools for the characterization of experimental samples in chemistry and materials science. XRD has been employed throughout industry and research laboratories for more than a century[1], and is well suited to characterize crystalline materials as it captures detailed information on the long-range periodic nature of crystal structures. NMR and Raman measurements, on the other hand, are more strongly dependent on localized chemical interactions and are widely used to characterize the structure of molecular materials[2,3]. Although their mechanisms and applications may differ, each of these characterization techniques produces similar one-dimensional spectra (sometimes referred to as *patterns*) that contain peaks with distinct positions, widths, and intensities. These features often serve as "fingerprints" for molecules and crystalline phases, which can be used to match unknown samples. Identification of unknown specimens can be accomplished by comparing newly measured spectra with those of previously reported materials in experimental databases such as the ICSD or RRUFF[4,5]. Nonetheless, experimental artifacts, such as measurement noise and background signals, as well as naturally occurring minor variations of the patterns, complicate the analysis process based on similarity. To automate this process, machine learning has recently emerged as an effective tool that can map experimental spectra onto known structures, with reported accuracies exceeding standard similarity-based metrics[6,7].

One popular method within the domain of machine learning is the artificial neural network, which stacks multiple layers of artificial neurons to resemble the structure and function of the human brain[8]. The application of neural networks for the use with spectroscopic patterns was first demonstrated in the work of Park et al.[9], where a convolutional neural network was trained to classify XRD patterns by their structural symmetries, distinguishing between different Bravais lattices and space groups. Later work extended the use of neural networks to identify particular phases from XRD patterns[10], even dealing with multi-phase mixtures[11–13]. For the analysis of NMR and Raman spectra, similar methods have also been used to assist manual analysis[14] and automate the identification of molecular species[15–17]. In each of these studies, a convolutional neural network structure was employed in order to enable the network to eliminate the experimental artifacts and facilitate the identification task, but every work developed its own distinct architecture.

The concept of convolutional neural networks (CNNs) was first introduced by LeCun et al.[18] for the task of classifying handwritten digits in images. These networks employ convolutional filters to recognize local patterns in the image, such as edges and textures, which are then combined to form a more abstract representation of the digit, independent of minor visual variations or positional changes in the image. Similarly, the relevant peak information in spectroscopic signals, another "local pattern", has to be identified and extracted from measured spectra that contain experimental artifacts, such as noise or background effects. It was not until the advent of improved computational capabilities in modern hardware and the introduction of large-scale image classification benchmarks, such as the ImageNet challenge[19], that the full potential of CNNs was realized. For the task of image classification, Krizhevsky and colleagues[20] demonstrated that a CNN could surpass the performance of other machine learning algorithms using manually tailored features, highlighting the general effectiveness of this approach.

Multiple studies have demonstrated the ability of CNNs to effectively eliminate experimental artifacts and correctly classify measured patterns, even when the network was trained on simulated data[10,12,13]. However, these studies often developed their own unique convolutional architectures and reported outstanding performance metrics on their specific datasets without comparing different architectures or systematically analyzing the general limitations of the presented networks. For example, Ho et al.[16] presented a CNN that could distinguish between 30 bacteria isolates from Raman spectra, reaching a

[1]Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Karlsruhe 76131, Germany. [2]Department of Materials Science & Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA. [3]Department of Materials Science & Engineering, UC Berkeley, Berkeley, CA 94720, USA. ✉email: markus.reischl@kit.edu

reported accuracy score of 82.2%. Similar methods have also been applied to X-ray diffraction, for which Lee et al.[11] reported a model that correctly identified 100% of crystalline phases in multi-phase samples. Though, it remains unclear whether the substantial disparity between the accuracies of these two models is due to their differences in neural network architectures, or due to the difficulty of the classification task for Raman spectroscopy versus X-ray diffraction. This question is further complicated by the introduction of several network architectures that have made use of more advanced deep learning techniques, such as VGG networks[10], Batch-Normalization[15,16], and inception[11] or residual blocks[16]. While these methods have been shown to be effective for the identification of complex features in images, their effectiveness has not yet been proven for the classification of one-dimensional spectroscopic data. Revealing the benefits of each neural network component is key to developing a single, universal deep learning model that can analyze spectra from various characterization techniques - similar to the introduction of general and transferable models that have been developed for the analysis of diverse image datasets.

To better understand the generalizability of neural networks as applied to spectroscopic analysis, more data is needed. There exist several large databases that contain experimental spectra obtained from a variety of materials and molecules. For example, the RRUFF provides a combination of XRD and Raman spectra for approximately 5800 minerals[5]. Similarly, NMRShiftDB provides NMR spectra for approximately 45,000 organic molecules[21]. In general, however, these databases cover only a small portion of the entire chemical space. The ICSD alone contains >260,000 crystal structures that have been experimentally synthesized[4], and many more hypothetical materials have been proposed[22]. In addition to the scarcity of experimentally measured spectroscopic data, the available spectra for a given compound may not accurately represent later measurements where sample artifacts and instrumental aberrations can lead to variations. For example, XRD patterns often show differences in peak positions and heights as a result of strain and preferred orientation in the corresponding sample. Likewise, changing the buffer solution used during NMR measurements can lead to large differences in the spectra obtained from identical molecules. To train machine learning models that are robust against such experimental complexities, all suspected variations should be taken into account when collecting the training data, which requires the development of new data sources that can be tailored to fit the problem at hand.

There are two distinct methods to organize a large and diverse dataset for evaluating the performance of various neural network architectures. One may gather experimental measurements on actual samples; however, obtaining sufficient data to train a robust neural network using such an approach would require an exceedingly large number of measurements performed on samples with varied compositions and artifacts[15,16]. Alternatively, one may rely on previously reported experimental entries (known crystal structures or molecular species) available in public databases (such as the ICSD) for the simulation of spectra that can then be augmented in a rapid and systematic fashion to account for possible artifacts[9,23]. This approach is particularly well suited for the generation of XRD patterns, which are readily calculated by applying a Fourier transform to a given crystal structure[12,13,24]. In contrast, the accurate simulation of Raman and NMR spectra requires more expensive ab initio calculations based on density functional theory, limiting the rate at which training data can be generated for machine learning[25,26]. Even then, curating a large and diverse dataset of entries that reflects the difficulties of manual analysis while ensuring that all distinct classes are theoretically separable remains a cumbersome task. Most databases contain duplicates or list several minor variations of nearly identical phases, and measurement errors sometimes lead to incorrect classifications of composition or structure. This problem is further complicated by the fact that different experimental techniques generally produce signals of different lengths. As such, disparate sets of data cannot be easily combined, despite their visual and characteristic similarities.

As an alternative, we introduce a universal synthetic dataset that can be used to train and validate machine learning models for the automatic classification of spectroscopic data. This simulated dataset contains features that are shared between distinct characterization techniques, thereby providing a representation of spectra obtained from XRD, Raman, or NMR measurements. Because our simulation algorithm does not rely on physics-based simulations, we are able to directly manipulate both the generation of unique spectra and the variances within patterns of identical classes, and it rapidly generates data with little computational cost. Our approach enables the examination of scenarios that may be challenging or infeasible to replicate in real-world environments or conventional simulation techniques. The resulting dataset is used to evaluate the performance of eight different neural network architectures that were previously developed for the classification of XRD and Raman spectra. Our tests reveal that, while all models perform relatively well on the synthetic dataset, they can occasionally fail to correctly classify spectra with overlapping peak positions or intensities. These results highlight several key weaknesses of neural networks and their application to spectroscopic data, and accordingly, we give several recommendations on how such models can be improved. All code and data discussed here is made publicly available, including the simulation algorithms and trained models, and we invite the community to build upon this repository by contributing further network structures and results.

## RESULTS

### Benchmark dataset

In this study, we aim to evaluate the performance of various machine learning models for classifying spectroscopic data. To achieve this, we propose a synthetic dataset that comprises 500 distinct classes. Here, each class is designed to represent a unique crystalline phase (XRD), chemical species (Raman), or molecule (NMR). As such, each class is characterized by a specific number of peaks (between 2 and 10), with distinct positions and intensities, allowing for unambiguous identification in its ideal, non-perturbed form. The classes are generated stochastically to produce a diverse range of spectra without inducing any bias in the number and density of peaks within each sample, minimizing the risk of obscuring unexpected challenges in the dataset. However, in a manner similar to that in the measured spectra, each class can have some variations in the positions and intensities of its peaks. Additionally, the shape of each peak can vary between each spectrum, reflecting certain properties of the experimental specimen as well as instrumental aberrations. Considering these possible artifacts, we arrange a dataset of 500 unique classes and simulate 60 randomly varied training samples per class, resulting in 30,000 total patterns. These were generated in less than 15 seconds using a standard desktop computer, demonstrating the speed at which synthetic data can be collected.

To prevent machine learning algorithms from simply memorizing the training samples associated with each class in our synthetic dataset (overfitting), we split the 60 patterns per class into 50 samples used for training and 10 for validation. Furthermore, an additional blind test set was also formed to accurately measure the performance of each model, since validation samples are used during the model selection process, possibly leading to information leakage[27]. Variations in the training and test data of a single class are illustrated in Fig. 1. The left panel shows three continuous spectra, including the ideal representation (blue) of the specified class, as well as two

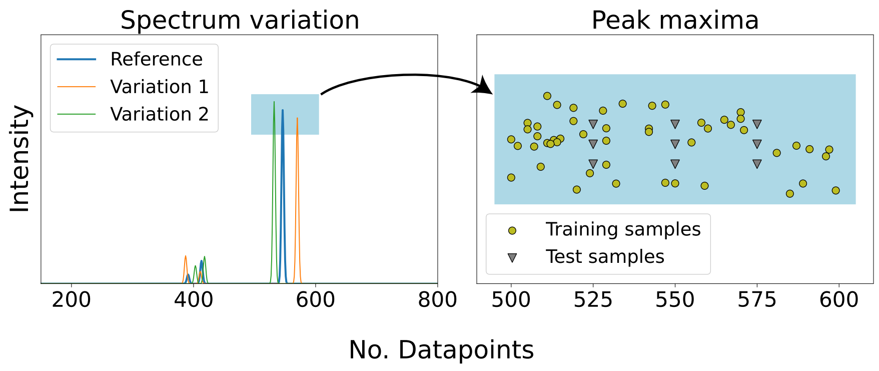

**Fig. 1 An illustration showing the variations included in our spectra dataset.** The left pane shows three varied spectra of the same class, where the blue graph exhibits the "ideal" representation of the class with a Gaussian peak shape. For the orange and green graph, the positions and intensities are slightly shifted to account for possible experimental artifacts, and the standard deviations of the Gaussians are also varied to broaden or narrow the peaks shapes. Second, we show the maximum value of the second peak in detail (as outlined by the opaque blue rectangle) in the right pane. Here, we show examples for the position and heights of randomly varied training samples (blue), as well as the uniformly varied test samples used for evaluation purposes.

randomly selected variations (green and orange). To better visualize all the variations that take place for this class, the right panel shows the distribution of discrete peak information (position and height of each peak maximum) from samples in the training and test sets. These correspond to variations in the second peak that is shown in the left panel, as highlighted by the blue-shaded region. The training samples (blue circles) are randomly distributed across a broad range of positions and heights, whereas the test samples (grey triangles) follow a uniform arrangement with less variation. To ensure that the test data variations lie in regions that are well represented by our training data, we intentionally include smaller variations in the test set samples than those included during training.

Following the generation of our synthetic dataset, we chose eight different neural networks that have developed in various domains but have never been compared against each other. These models have distinct architectures, which can therefore be used to probe the influence of each network component on the performance of the resulting model in terms of classification accuracy. All networks make use of the convolutional neural network structure, which became popular for the classification of images. Such networks stack convolutional layers for the extraction of local features, pooling layers for the identification of features with distinct sizes, and fully-connected layers for the recognition of long-range correlations. Here, three basic convolutional neural network architectures were considered following the work of Lee et al.[11] and Szymanski et al.[13] (denoted CNN2, CNN3, and CNN6), which were developed for the analysis of multi-phase XRD patterns and each contain varying numbers of convolutional-pooling blocks. Two additional networks from the work of Lee et al.[11], which add a varied number of inception blocks to the conventional CNN architecture (denoted INC3 and INC6), are also considered. With a further increase in complexity, we consider the network of Wang et al.[10] (denoted VGG), which stacks multiple convolutional layers before pooling operations to extract more complex features and was proposed for the identification of single-phase XRD patterns. Lastly, to complement the neural networks that we designed for the analysis of XRD data, we also include two models that were developed for the analysis of data from Raman spectroscopy. These are denoted CNN BN[15] and Resnet[16], which employ Batch-Normalization and residual blocks, respectively. These types of architectures were originally proposed to improve the interpretation of images, but have not yet been tested in depth for the classification of one-dimensional spectroscopic data. In addition to their differences in convolutional and related layers, the models considered here also differ in terms of their use of fully-connected layers (FC) and activation functions. The detailed information associated with the architecture of each model is presented in Table 1. We note that all models share the same output layer, which contains 500 neurons that correspond to the number of classes in our synthetic dataset.

**Table 1.** A description of the neural network architectures tested in this work.

| Model Name | Publication | Architecture | FC Neurons | FC Activation |
|---|---|---|---|---|
| CNN2 | 11 | (C-MP)x2 | 2000-500 | Linear |
| CNN3 | 11 | (C-MP)x3 | 2500-1000 | Linear |
| CNN6 | 13 | (C-MP)x6 | 3100-1200 | ReLU |
| VGG | 10 | C-MP-(C-C-MP) x3 | 120-84-186 | Linear |
| CNN BN | 15 | (C-BN-MP)x3 | 2048 | Linear |
| Resnet | 16 | C-BN-RESx6 | – | – |
| INC3 | 11 | (C-MP)x3-INCx3 | 3700-740 | Linear |
| INC6 | 11 | (C-MP)x3-INCx6 | 4000-400 | Linear |

The following abbreviations are used: *C* for 1D convolutional layers, *MP* for 1D Max Pooling operations, *FC* for fully-connected layers, *BN* for Batch-Normalization, and *ReLU* for the rectified linear unit activation. More complex structures, such as residual (RES) and inception (INC) blocks, are also abbreviated here. Full details, such as kernel sizes for convolutions, can be found in their respective publications, as well as in our repository jschuetzke/synthetic-spectra-benchmark/model_implementations.

With the exception of CNN6, most networks that have been developed for the analysis of spectroscopic data do not utilize non-linear activation functions between their fully connected layers (see Table 1). While the Resnet model completely omits fully-connected layers, the networks introduced by Wang et al.[10], Lee et al.[11] and Liu et al.[15] use multiple fully-connected layers without any non-linearities. This is somewhat surprising, as stacking multiple fully-connected layers without using a non-linear activation function between them provides identical capabilities as a single fully-connected layer, and therefore the resulting model is equivalent to a linear classifier. We note that a linear model operates by learning appropriate weights for all datapoints and calculating their weighted sum, possibly with an added bias, which is equivalent to learning a single threshold for the classification of a given signal. Because spectra typically contain more than one peak, we expect that a robust classification model needs to learn multiple thresholds to distinguish phases with similar peak positions and intensities. As shown in Table 1, CNN6 is the only model with such capability as it uses a rectified linear unit activation (ReLU) between its fully-connected layers, and therefore the network can learn different thresholds for every neuron in each layer.

## Benchmark results

Using our synthetic dataset that contains 500 classes, we trained each of the aforementioned models five times, applied them to the classification of samples in the test set, and recorded their accuracies. The convergence of neural networks is known to be highly dependent on the initialization of their weights, and therefore we trained multiple models with distinct initializations to prevent conclusions being drawn from unfavorable starting conditions. We use identical training conditions for every model (batch size 128, Adam optimizer with a learning rate of $3*10^{-4}$) and monitor the performance on the validation samples to reduce the learning rate once the convergence stagnates, plus we stopped the training process when overfitting started to occur (early stopping). The final model for each training run was selected by choosing the one that led to maximal validation accuracy.

In Fig. 2, we present the number of misclassifications made by each model when applied to the test set. Full details on each model's performance are given in Supplementary Table 1. All models are found to achieve a high accuracy on our synthetic dataset, as each correctly predicts at least 4450 samples from a total of 4500 samples in the test set (98.9% accuracy). These results show that despite being developed for completely different domains (e.g., XRD versus Raman spectroscopy), the models perform similarly well when applied to spectra with shared features from each characterization technique. As shown in Supplementary Fig. 1, selected pairs of synthetic spectra classes that exhibit similar patterns were still all correctly classified. Neither spectra with a high number of peaks, nor spectra, where some of the peak positions or intensities are identical, cause a misclassification, as long as further information is present to accurately separate the spectra pairs. This highlights that spectra are generally easy to classify, consistent with the high accuracy scores reported in other publications.

Yet, there still do exist some differences in the accuracies of the models considered here, warranting further investigation. In particular, the CNN6 model developed by Szymanski et al.[13] achieves the best performance on the test set, producing only eight misclassifications on average, out of 4500 samples in total. By contrast, most models misclassify between 10 and 40 samples depending on their initialization, with the exception of CNN BN, whose performance ranks worse than all other models and shows a wide range of performance variation between training runs.

Considering only the convolutional architectures, we find little correlation between the number of convolutional layers and the accuracy of the corresponding model. Even though CNN3 performs better than CNN2 and has one more convolutional layer while the remaining architecture is similar, the performance enhancement does not hold for more complex models. For example, the VGG and Resnet models do not substantially improve performance over standard CNN models, despite having many more convolutional layers. Although the CNN BN model has the same number of convolutional layers as CNN3, the accuracy of the Batch-Normalization architecture is remarkably worse. Contrary to the absolute number of convolutional layers, the inclusion of kernels of different sizes in the inception blocks of INC3 and INC6 yielded considerably better results than the CNN2 and CNN3 models with similar implementations. The overall winner of our synthetic benchmark (CNN6) may differ in its convolutional architecture (six convolutional layers plus pooling) from all other networks, but is also unique due to its use of a non-linear activation function in the fully-connected layers, so the exact contribution of the convolutional architecture versus the use of non-linearities remains questionable.

While the accuracy and misclassification metrics are important indicators of the final performance of each model, the training process also provides insights into the stability and convergence of the training process. Figure 3 displays the progression of the

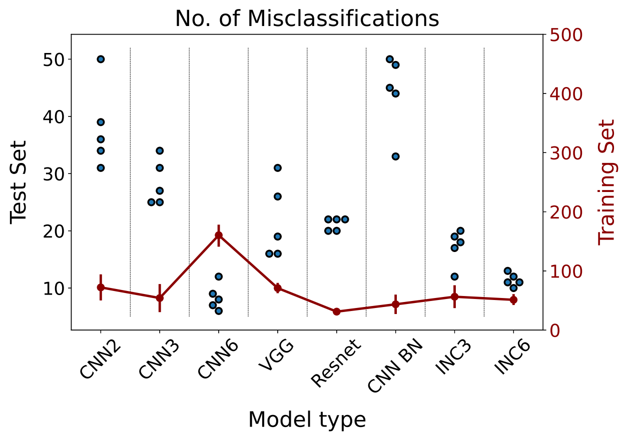

**Fig. 2 The performance of eight previously published models on our synthetic dataset is illustrated by the number of classifications each model makes on the test set (left axis, blue dots) and training set (right axis, red lines).** Each model is trained 5 times with different starting initializations (random seeds). Accordingly, the spread of the dots and error bars for the lines show the range of misclassifications made by each set of models.

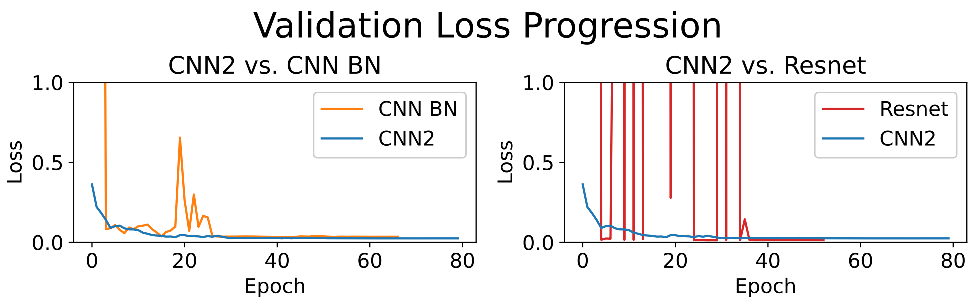

**Fig. 3 Comparison between the validation loss progression of models CNN2 (blue), CNN BN (orange), and resnet (red) during the training.** While the CNN2 model shows a typical convergence that minimizes the validation loss, the training process of the CNN BN and Resnet models illustrates unstable behavior. For the Resnet model, the validation loss is considerably higher for some epochs (ranging between 0 and 50), exceeding our defined limits of the Figure. Still, the Resnet model ultimately achieved a lower validation loss than the CNN2 model.

validation loss for different models throughout training (as a function of the number of epochs). Most models exhibited a stable convergence process, with a high loss at the initial epoch that is gradually minimized over time, as demonstrated by the blue graph of CNN2 in Fig. 3. However, the two models that incorporate Batch-Normalization layers in their architecture (CNN BN and ResNet) exhibit markedly different behavior, with their validation loss fluctuating throughout the training process. In the case of the Resnet model (red), the validation loss occasionally exceeds the limits of the displayed range in Fig. 3, but ultimately settled at a lower value than the CNN2 model, which is further reflected by the lower misclassification numbers in Fig. 2. Thus, the use of Batch-Normalization resulted in an unstable convergence process, but it did not impede the models from achieving fair performance results, so long as training was halted at a point of low validation loss.

Further insight into the performance of each model on the test data can be gained by considering their accuracy with respect to the training data, which are shown by the red points in Fig. 2. While most models correctly identified at least 24,900 of the 25,000 training samples, the CNN6 model exhibited substantially poorer performance on the training set, with between 125 and 185 training samples being misclassified. The disparity between test and training set performance, however, highlights the necessity of utilizing regularization methods during the training, such as dropout or stopping the training once the validation data performance does not improve anymore. While the Resnet model achieved the best performance on the training set, it did not include fully-connected layers following the convolutional stage and, therefore, also did not incorporate the dropout regularization method, which was present in every other network. This likely explains the comparatively poorer performance of the Resnet model on the test data. A different regularization strategy was employed by the VGG model, which applied dropout between the convolutional layers rather than the fully-connected layers. However, while this form of dropout prevented the VGG model from overfitting on the training data, it still only achieved an average classification result on the test set. The greatest disparity between training and test set performance was observed for the CNN6 model, which also employed the highest level of regularization, dropping out 70% of neurons, compared to the range of 20–50% employed by other models. Further investigation is required to understand the specific elements of the CNN6 model's architecture that enable it to generalize well to unseen samples, despite misclassifying more training set samples than other networks.

## Misclassification causes

To pinpoint the challenges associated with spectroscopic analysis, we analyze the shared features of spectra that were misclassified by the machine learning models studied here. Although the spectra generated in this work have a limited number of peaks per class (≤10), some classes display a high density of peaks within a narrow region. Yet, so long as these peaks do not overlap with those of other classes, the classification models are robust. This suggests that the neural networks tested here would remain accurate even on spectra with many more peaks, which are often obtained from measurements on complex samples (e.g., low-symmetry phases in XRD). Thus, the presence (or absence) of such peaks aids in distinguishing similar classes where other peaks may overlap. As will be discussed in the next three sections, all the misclassifications observed here can be attributed to one of three factors: (1) overlooking minor peaks, (2) overlapping peak positions, and (3) overlapping peak intensities.

*Overlooking minor peaks*. There are many cases where two classes differ by only a single peak. However, we find that when

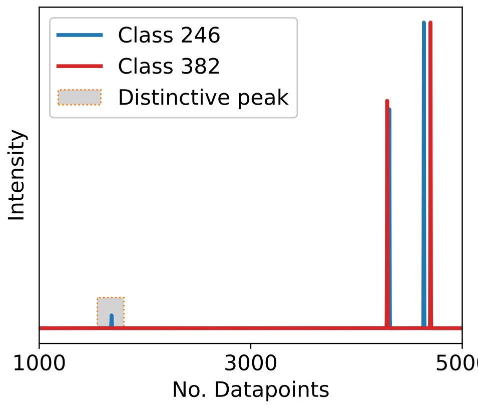

**Fig. 4 Simulated spectra are shown for classes 246 & 382.** These two classes share two large peaks that overlap. However, class 246 has an additional minor peak that is highlighted by the shaded region. All machine learning models overlook this minor peak and fail to distinguish the two phases.

this distinctive peak has a low intensity relative to the others, it is often overlooked by the neural networks. Such a case is displayed in Fig. 4, which shows the spectra associated with two different classes (246 and 382) that have similar peak positions and intensities for their two largest peaks. Class 246 (blue) can, in principle, be distinguished from class 382 (red) by a minor third peak. Nonetheless, all of the models tested here fail to consistently distinguish these two classes and instead predict a single class (382) for most samples in the test set.

Upon further investigation of each model's behavior, we found that the neural networks tend to overprioritize the position of the largest peak in a given spectrum when making a classification. In fact, the spectra associated with class 246 were correctly classified only when its largest peak was sufficiently shifted away from the position of the nearby peak in class 382. In addition to this current pair of classes (246 and 382), we found six additional pairs that caused misclassifications for the same reason. On the basis of these results, we suspect that neural networks struggle to learn sufficiently large weights associated with minor peaks in spectroscopic data, limiting their performance when the presence (or absence) of such peaks is the only means of distinguishing two classes.

*Overlapping peak positions*. Misclassifications are also found to occur on classes that contain an equal number of peaks with matching intensities and only slightly different positions. An example of this challenge is shown in Fig. 5, which depicts the distribution of positions and intensities associated with the training and test data for two classes (8 and 318). For simplicity, these properties are visualized for only one peak that should be sufficient to distinguish the two classes. Although their intensities are nearly equal, the positions of their peaks are different enough so that a clear decision boundary can be drawn to separate the test samples of the classes without any error (dashed line).

Nevertheless, all the machine learning models tested in this work struggle to correctly classify these two classes. In particular, those samples close to the decision boundary tend to lead to the highest number of misclassifications. This limited accuracy holds true even for the best model reported (CNN6), which incorrectly classifies three samples in the test set between these two classes. Most misclassifications occur when the test samples are located right next to a training sample (position- and intensity-wise) of the opposite class. In this case, the model rather memorized the positions of the training data than learning the general classification rules that apply to unseen data, which implies overfitting. We

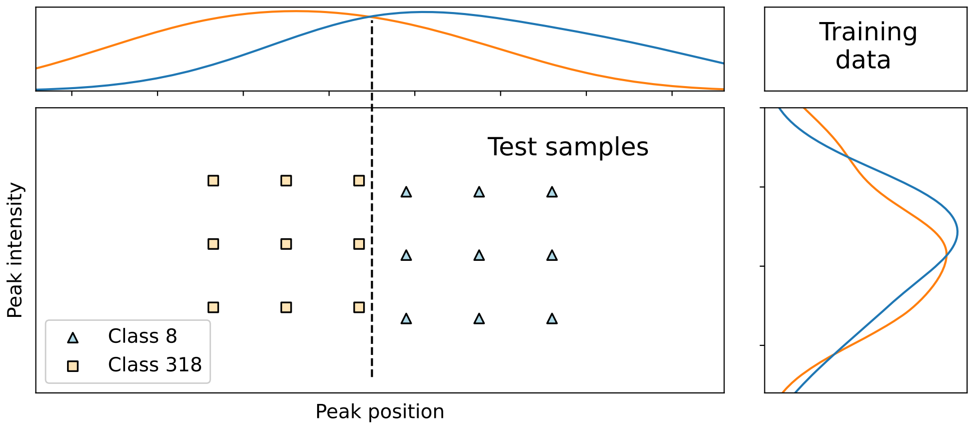

**Fig. 5 Position and intensity of decisive peak to separate between classes 8 (blue) and 318 (orange).** Top and Right show the distribution of the training data position and intensities of both classes. The peak intensity distributions of the training samples align almost perfectly and yield no meaningful information for a robust classification. Similarly, the distribution of the training peak positions overlap but there are areas that are more probable for either class. In the Center, we visualize in a scatter plot the test samples that align with the training data distributions. A robust classifier should correctly predict all test patterns, since their positions lie in distinct areas, yet most models tested here fail to make a correct distinction.

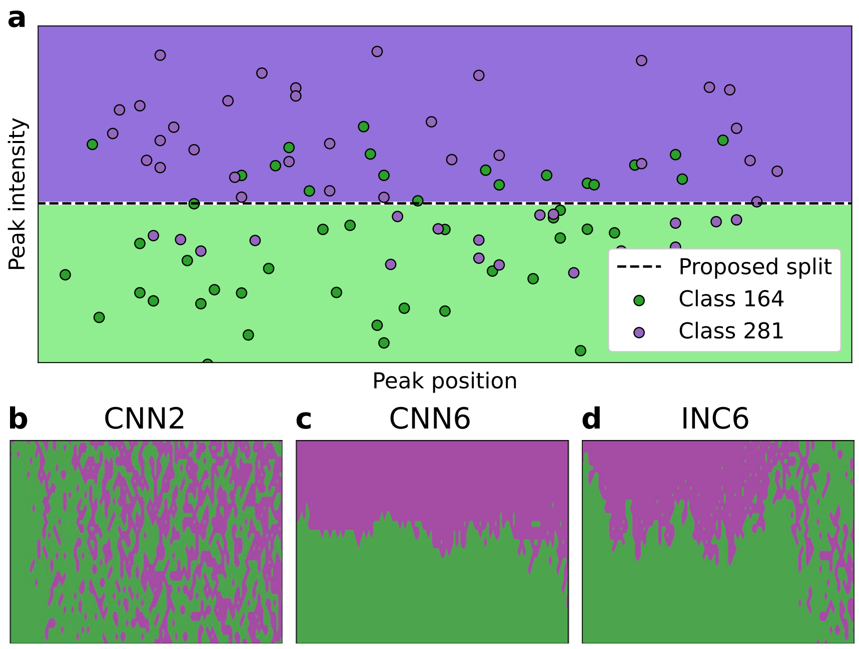

**Fig. 6 Comparison of the relevant peak positions and intensities to separate the samples of classes 164 & 281. a** Shows the distribution of training data samples of both classes, together with a proposed, linear split for a robust distinction, which assigns the lower intensity samples to class 164 (green) and the higher intensity samples to class 281 (purple). Using the linear split, all test samples can be predicted correctly. **b–d** Show the classification regions of models CNN2, CNN6 and INC6, respectively.

suspect that the inability of this model to decouple the nearby peaks can be traced back to its application of pooling, which reduces the dimensionality of the input spectrum. The resulting loss of resolution may cause two peaks with slightly different positions to appear identical, therefore making it impossible to distinguish the corresponding classes.

*Overlapping peak intensities*. There are several cases where peak intensities, as opposed to positions, are the only features that may be used to distinguish two similar classes. To visualize such cases, we present in Fig. 6a the distributions of intensities and positions of the major peaks associated with classes 164 and 281. While the positions of these peaks are identical and, therefore, cannot be used to distinguish the two classes, the intensities are sufficiently different such that a decision boundary (dashed line) can be drawn to achieve good accuracy on the test set. We find that the ability of machine learning to identify an optimal decision boundary is highly dependent on the architecture of the model.

To visualize this, the decision boundaries for classification made by three different models (CNN2, CNN6, and INC6) are shown in Fig. 6b–d. Predictions for class 164 are shown by the green regions, whereas predictions for class 281 are shown in purple. These plots reveal that the predictions made by CNN2 appear to have no correlation with the intensity of the peak. Instead, each green and purple regions form vertical stripes, which suggests that the model (unsuccessfully) attempts to distinguish classes 164 and 281 by their peak positions, possibly due to overfitting on the samples in the training set. On the contrary, CNN6 learns to distinguish the two classes by their peak intensities as desired, proposing a decision boundary that appears similar to the optimal

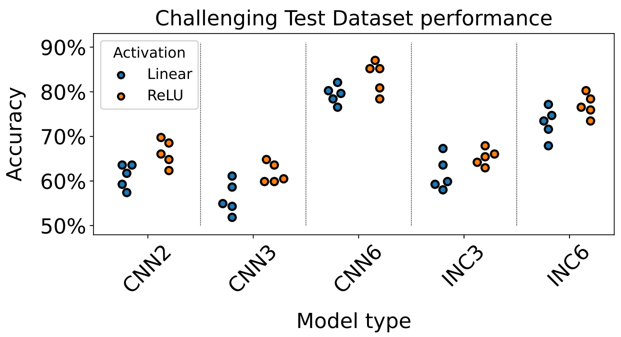

**Fig. 7 Investigating the activation function impact on the challenging test dataset.** For every model, we trained 5 initialization using a linear activation function for the fully-connected layers and compared the predicted accuracies on the test set in comparison to variants trained with a ReLU activation function. Blue shows the variation for the linear activation variants, and orange illustrates the accuracy distribution for models with non-linearities. The performance improves for all model variants with the ReLU activation function.

boundary shown in Fig. 6a. The last model, INC6, represents an intermediate case between the two extremes illustrated by CNN2 and CNN6, as it appears to learn some intensity threshold but with a large degree of overlap between classes 164 and 281. This overlap prevents the model from achieving high classification accuracy on the corresponding test samples.

The success of CNN6 in learning an optimal decision boundary that can separate peaks by their intensities highlights the importance of including non-linearities between the fully-connected layers of the neural networks. Since the fully-connected layers treat every single datapoint independently, the network has to learn multiple thresholds in order to separate two classes according to their peak intensities alone. In contrast to the other models tested here, CNN6 features a ReLU activation function between its fully-connected layers and therefore is able to learn non-linear classification rules. This is supported by the classification map shown in Fig. 6c, where the decision boundary coincides well with the optimal one.

## Challenging test dataset

While the previous synthetic dataset was generated randomly to investigate the classification limitations of each model on data generated without any inductive bias, we next created a more challenging set of spectra to further distinguish the performance of each model. Here we employed a similar approach as before to generate training and validation samples, except now the underlying classes were specifically tailored to exhibit a substantial degree of peak overlap (both in terms of positions and intensities). In total, the challenging test dataset contains 27 classes where three groups of 9 were tailored to fall under one misclassification category (detailed in the previous sections). The ideal representations of these classes are visualized in Supplementary Fig. 2.

Using our generated dataset, we re-trained all the neural network architectures in two separate forms: with linear activation or with ReLU activation applied between their fully connected layers. This enables us to probe the influence of the activation function on the performance of each model and confirm its importance for accurate classification. Figure 7 shows the test set accuracies for the models trained with linear activation (blue) versus ReLU (orange). The overall performance of the models improved considerably when using the ReLU activation function, which confirms our hypothesis that non-linearities are essential for the classification of spectra with overlapping peaks. The CNN6

**Table 2.** Median test set accuracies for select modified network architectures and their original counterparts on the challenging test dataset.

| Implementation | CNN BN | Resnet | CNN6 BN* | VGG | CNN6 ConvDO* |
|---|---|---|---|---|---|
| Original | 52.47% | 3.70% | 67.90% | 68.52% | 79.63% |
| Modified | 58.64% | 80.74% | **85.19**% | 83.95% | **85.19**% |

The performance improves for the modified versions that omit the Batch-Normalization layers or move the dropout regularization from the convolutional stage to the fully-connected layers. The highest attained metric is emphasized using bold formatting.
*The original implementation of the CNN6 BN and ConvDO refers to the variant with activate Batch-Normalization or convolutional dropout methods, while the modified version indicates the previously reported CNN6 performance without changes to the established CNN6 implementation.

model still achieved the best performance with a test set accuracy close to 90%, reinforcing our previous findings on the general benchmark. In contrast, even with the use of a non-linear activation function, the INC3 and INC6 models both performed worse than the CNN6 model, calling into question the necessity of the inception blocks. Notably, the CNN2 model performed better than the CNN3 model on our more challenging dataset and even caught up with the performance of the INC3 model, further supporting our argument that the inception blocks are not beneficial for the classification of spectra.

In our study, we found that three models were not able to converge effectively, even when using the ReLU activation function. The Resnet specifically could not utilize the updated activation function given that it does not include any fully-connected layers. While the Resnet achieved average results on the general benchmark, it completely failed to converge on the challenging test dataset and as a result produced accuracy metrics that were equal to those achieved by random prediction. Similarly, the CNN BN network showed little improvement from the modified activation function, as the rescaling of patterns performed by the Batch-Normalization between convolutional layers may have led to some information loss. Lastly, the VGG network performed poorly with either activation function since it applies dropout between the convolutional layers, potentially altering the appearance of the training spectra by excluding certain peaks.

In order to systematically evaluate the impact of these methods, we modified the CNN BN, VGG, and Resnet networks and re-trained the adapted models on the challenging test dataset. In the modified networks, we removed the Batch-Normalization or convolutional dropout layers, but incorporated fully-connected layers with ReLU activations for the modified Resnet model. We further investigated the impact of these methods on the convergence of the training by adding Batch-Normalization (CNN6 BN) or dropout between the convolutional layers (CNN6 ConvDO) of the CNN6 model. While we adhered to the previously established training methodology and trained each model using 5 different initializations, we report only the median scores for the different model implementations in Table 2. The top row of the results shows the performance metrics for the original implementations of each model, while the bottom row presents the accuracy scores for the modified models without the special methods. However, we found that in all cases, the simpler models without Batch-Normalization or dropout between the convolutional layers performed best. As previously observed for the general benchmark, active Batch-Normalization layers resulted in an unstable validation loss, which is illustrated in Supplementary

Fig. 3. Notably, the best performance was still achieved by the CNN6 model, which further suggests that neither the VGG-architecture nor the residual blocks improve the classification of the synthetic spectra.

## DISCUSSION

Many different neural network architectures have been developed for the analysis of characterization data from techniques such as XRD, NMR, and Raman spectroscopy. However, the models presented for different techniques have never been tested across domains, despite the visual similarities of the spectra. To benchmark the different neural network architectures, we have provided a method to rapidly simulate artificial spectra with user-defined features and generated a dataset that depicts common artifacts related to changes in peak positions, widths, and heights. Instead of relying on manually curated entries from a database, which may introduce bias in the selection of certain compositions or structures, our approach is “structure free” in that it generates spectra from stochastically chosen peak positions and intensities. This allows us to probe specific situations which would otherwise be difficult to reproduce in experimental settings.

On this dataset, we trained eight distinct network architectures, and each model achieved a high classification accuracy (≥98.9%), while no clear differences were observed between the performance of models developed for the analysis of XRD versus Raman spectroscopy. The high level of accuracy achieved by all the networks tested here suggest that spectra are generally less challenging to classify than images, at least when considering only one phase per spectrum. Indeed, spectral analysis does not require the encoding of two-dimensional features (such as “eyes” or “ears”) which would otherwise need to be learned by neural networks when applied to images. Instead, accurately classifying distinct spectra only requires that the networks learn appropriate thresholds that can be used to distinguish various classes by their peak positions and intensities. Although this task appears to be successfully accomplished by many different neural network architectures, we caution that a high accuracy metric does not necessarily imply generalizability on more challenging cases involving classes with significant peak overlap. This is shown by our tests performed on the challenging test dataset, where certain networks significantly outperform others despite their similar level of accuracy on the original dataset.

Thus, although all models produced high accuracy on the synthetic dataset, some architectures performed better than others on our universal synthetic dataset containing a wide range of scenarios. In particular, the CNN6 model developed by Szymanski et al.[13] produced the least number of misclassifications on both our general and complex datasets. We attribute the improved performance of this model to two key factors. First, the use of a non-linear activation function (ReLU) between the fully-connected layers in the neural network ensures that multiple thresholds can be learned to distinguish similar classes. As shown in Fig. 6, this enables the identification of a clear decision boundary to separate spectra by their peak intensities, whereas models that use only linear activation functions fail to learn such a boundary. Second, the high rate of dropout regularization employed throughout the training process ensures that the model does not overfit the training data. Although most other models tested here also use dropout between their fully-connected layers, albeit at a lower rate than CNN6, the performance gap between training and test samples implies that overfitting occurs. Alternatively, regularization methods that restrict the magnitude of weights might be useful to prevent the networks from memorizing training samples.

Our tests reveal that more sophisticated architectures, such as VGG, residual, or inception blocks, do not provide improved accuracy in the classification of spectroscopic data. Upon evaluating both the original implementations and their modified variants incorporating ReLU activation units or other measures, it was observed that the more complex models demonstrated inferior performance when compared to the simpler CNN6 model. This result may appear somewhat surprising at first, as such models have been widely employed to analyze images more accurately; however, the spectra considered in this work have different properties than images. The convolutional filters utilized in the networks are specifically designed to attend to a restricted portion of the input and are systematically shifted across the image to identify complex local patterns of interest, regardless of the absolute position of the input. In contrast, our results suggest that the identification of local information, such as peaks or clusters of peaks, requires far fewer convolutional layers than is necessary for the classification of images. On a related note, the use of Batch-Normalization, which is a commonly applied method to achieve robust convergence of image classification networks, actually caused an unstable training process for all of the models tested here. Batch-Normalization is typically employed to shift and rescale the locally extracted features of convolutional layers, so it possibly alters the relative peak intensities of the spectra, complicating the classification process. As the relative intensity for spectra is zero for all ranges without peaks, the mean and standard deviation of the whole pattern is mostly affected by the width of peaks or altered peak intensities. Both of these features vary considerably from sample to sample, so the normalization process diversifies the intensities relevant for the subsequent layers.

Despite its overall promising performance, CNN6 still shows some limitations. For example, this model, as well as the other neural networks, consistently overprioritize the largest peaks in each spectrum. While this leads to the correct classification in most cases, it can occasionally cause misclassifications on spectra where the largest peaks overlap and only the smaller peaks can be used to distinguish similar classes. This problem is particularly difficult to resolve as larger weights must be learned to enable the smaller peaks to have a significant influence on the final classification of the neural network. Yet, larger weights are often associated with overfitting and accordingly are penalized when using regularization. The type of magnitude of the regularization technique used should therefore be considered as an additional factor to be optimized during model selection. Similarly, CNN6 fails when two peaks occupy positions close to each other, as shown in Fig. 5, since numerous pooling operations are applied throughout the convolutional stage, which lower the resolution of the patterns.

To facilitate the development of improved machine learning models for spectroscopic analysis, we make our code and data publicly available. This includes not only our existing synthetic dataset that was used to benchmark the performance of all models tested here, but also the scripts that were used to generate the corresponding data. As detailed in the methods, these scripts allow the user to customize various aspects of the dataset including the number of classes it contains, the abundance of peaks in each spectrum, and the range of variations associated with these peaks. We note that all spectra are kept relatively simple, with Gaussian peak shapes and no background signal, as this study is focused on the general ability of neural networks to distinguish spectra. Using our code and its associated data, we invite researchers to develop and test new models that can outperform the existing methods presented in this work. As the ImageNet challenge sparked innovation for the analysis of image data with more sophisticated architectures such as Resnet, we believe that a universal synthetic dataset will enable more reliable model validation across all domains of spectroscopic analysis.

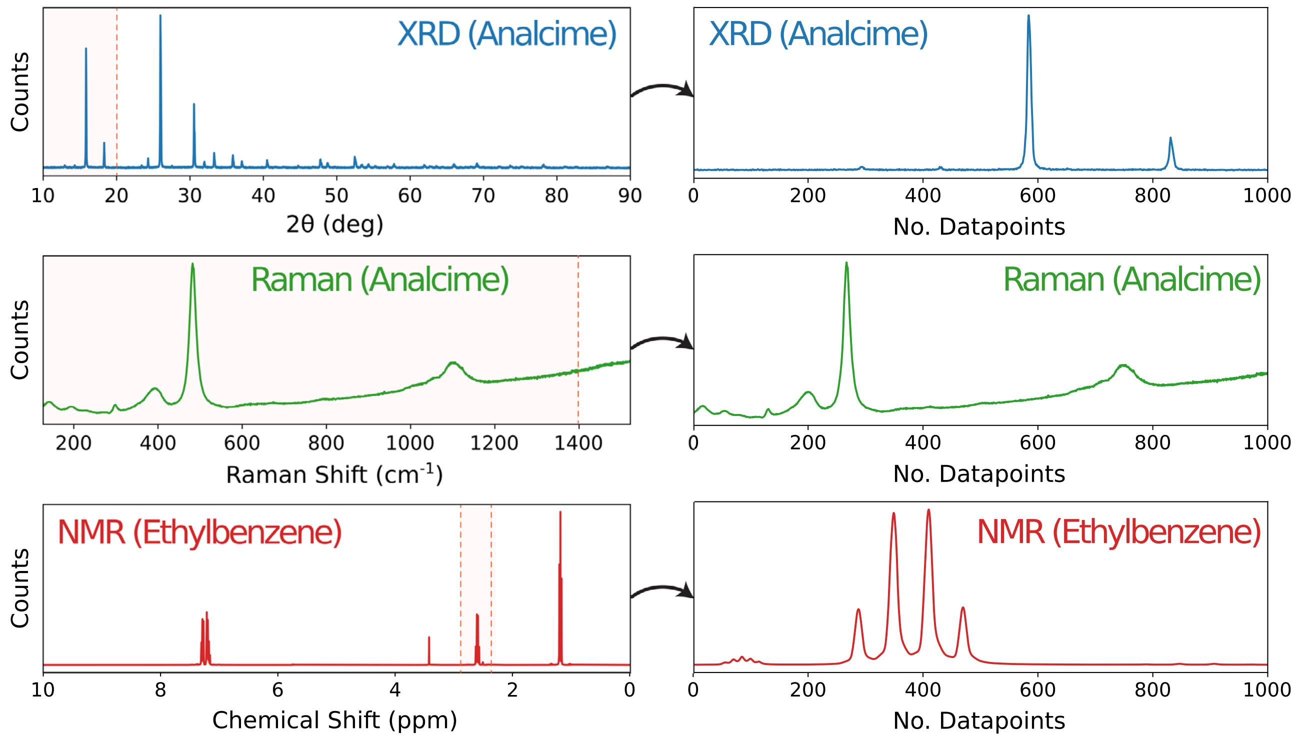

**Fig. 8 Three examples of spectral data are shown.** XRD and Raman patterns are given for Analcime, a common inorganic mineral[5]. An NMR spectrum is presented for Ethlybenzene, a well-known organic compound[30]. The left panels shows raw data spanning the entire measurement range. The right panel shows the same spectral data, now standardized to contain 1000 measurement datapoints. The corresponding ranges are highlighted (orange) in the left panel.

## METHODS

### Synthetic dataset approach

In order to generate a synthetic benchmark set that can be used to evaluate machine learning models across different domains, the spectra have to represent common properties found in patterns from the various sources. Figure 8 shows three exemplary spectra obtained from XRD, Raman, and NMR measurements on samples of Analcime and Ethylbenzene. As shown in the left panel of Fig. 8, spectra obtained using different techniques appear quite different at first glance. For example, Raman spectra often contain fewer and more diffuse peaks than XRD of NMR patterns. However, upon further inspection, the density of peaks with respect to the sampling of datapoints remains relatively consistent between different techniques. The Raman pattern shown in Fig. 8 contains only 1000 distinct datapoints, whereas the XRD and NMR patterns contain 8000 and 65,000 datapoints, respectively. If we instead crop each of the spectra to contain only 1000 datapoints, as shown in the right panel of Fig. 8, the resulting patterns appear much more similar. Regardless of which technique is used, the normalized spectra each contain 2–6 peaks with comparable widths. Therefore, we conclude that each technique produces data with similar "information density," justifying our choice to use a single artificial dataset for testing and validation of machine learning models.

For all characterization techniques considered here, experimental artifacts and instrumental aberrations can cause spectra to deviate from their expected fingerprints. For example, peak intensities from XRD scans often show large changes caused by a non-random orientation of particles in the specimen. Similarly, the positions of peaks in NMR spectra can be shifted by the choice of the buffer solution used during the measurement. To account for such effects, our synthetic dataset introduces stochastic variations related to changes in peak positions, intensities, and widths. However, these variations should be kept minimal to avoid unwanted overlap between specimen with similar fingerprints. Experimentally, the reduction of artifacts can be accomplished through careful sample preparation (e.g., grinding of powder samples for XRD).

We specify multiple parameters for our synthetic dataset approach that can be changed to fit the specific use-case. For example, every spectrum should contain an equal number of datapoints that span a fixed range (i.e., equal start and end points). Additionally, there should be a fixed number of possible classes used to generate distinct spectra, as these classes will be used to train machine learning models later on. Accordingly, we define a set of universal parameters for the synthetic dataset as follows:

- The number of datapoints contained in each pattern
- The number unique classes in the dataset (ideal patterns)
- The minimum and maximum number of peaks in each pattern

In contrast to experimental scans, our synthetic spectra do not have a real measurement range but are instead described by the number of datapoints contained by each spectrum. To simulate spectra that are representative of experimental data (i.e., similar to the spectra shown in Fig. 8), we recommend that parameters are chosen to produce a comparable "information density", e.g., with 2 to 6 peaks per 1000 datapoints. The parameters can be manually set by the user to fit the problem at hand in (dataset_config_generator). Based on the settings, we simulate classes by randomly sampling different peak positions and relative intensities. Following the execution of the generator script, each class is defined by a unique set of discrete peak positions and intensities. This class information is stored in single file describing the entire dataset.

We next generate spectra from the ideal class representations by adding minor alterations that account for possible experimental artifacts. This method was first introduced by the work of

Oviedo et al.[23] for X-ray diffraction and was later extended to the field of Raman spectroscopy by Rui et al.[28]. Such methods have been shown to result in robust classification networks that perform well on experimentally measured data, even though they have only been trained on simulated data[11–13]. Here we utilize a similar approach to generate simulated variations for training, validation, and test samples based on stochastically chosen peak position and intensity shifts, which are fit to Gaussian curves to produce continuous spectra. Furthermore, our method does not explicitly account for noisy or diffuse background signals as previous studies have shown that the models considered here are robust against such artifacts. In cases where the background signal is substantial, the main effect is the modification of peak intensities, for which variations are already included in our dataset.

This process relies on the following parameters:

- The number of augmented patterns generated for each class
- The magnitude of applied variations:
  - Maximum peak position shift,
  - Maximum peak intensity change,
  - Range of Gaussian peak widths.

Here, all variations are applied independently of one another. This differs from experimental artifacts, where changes are often coupled. For example, strain-induced changes in the peak positions of XRD spectra are known to follow a well-defined relation. Moreover, peak shapes are often more complex than simple Gaussian bell curves (e.g., Voigt profiles), and the widths of these peaks can be correlated with one another throughout a given spectrum. However, our current approach is designed with simplicity and generalizability in mind. The resulting data should still be sufficient for the validation of machine learning models, which can be applied to experimental data later on. A key advantage of simulations is their ability to rapidly generate customized synthetic datasets, which can be used to test classification algorithms with respect to different features and variations. For example, one can evaluate how many randomly varied samples are necessary per class to train a robust classification model given certain parameters (number of data-points, peak counts, and degree of overlap).

By testing various configurations of our dataset generator, we identified a set of parameters that include a large number of classes (500) and only slight overlap between similar spectra when variations are applied. With these parameters, each pattern is comprised of 5000 datapoints that contain 2–7 peaks. We also impose limits on all peak positions such that they maintain a distance of at least 100 datapoints from the borders of the spectrum. This ensures that no peaks are lost when their positions are varied. Based on the parameters used here, a json file is generated containing the information associated with each class. This file can be found in our repository: dataset_configs/benchmark.json.

## Neural network architectures

Spectroscopic data can be classified in a number of ways. For example, several models have been developed to categorize the space group of crystalline materials from XRD patterns, thereby grouping completely different patterns into a single class defined by symmetry relations among peaks[9,23,29]. In contrast, it is more common to use spectral data for the identification of distinct classes by comparing the positions and heights of observed peaks with known reference data. This is the task that we consider in this work, which may be framed as a one-to-one mapping between each “fingerprint" and class.

The first deep learning model developed to classify spectroscopic data, introduced by Liu et al.[15], demonstrated the successful application of a neural network on various mineralogical datasets of Raman spectra. Ho et al.[16] later refined this network for identifying Raman spectra by incorporating residual blocks, commonly used in advanced image classification models. However, the evaluation was performed using a different dataset, classifying bacteria isotopes instead of crystalline materials, and no direct comparison of performance between the networks was presented. In parallel, Wang et al.[10] presented the first neural network for identifying crystalline phases from XRD patterns, utilizing the VGG architecture, which also originated from the field of image recognition. Similarly, Lee et al.[11] advanced the analysis of XRD patterns by developing a network that could not only classify single-phase patterns, but could also be applied to samples containing multiple phases. Szymanski et al.[13] further improved the performance of deep learning models for the classification of multi-phase samples by iterating between the identification and subtraction of single-phase components. While each of the previously reported methods appear to be effective on specific domains, a direct comparison of the various network architectures has not yet been made. To better understand the advantages and limitations of each model therefore requires that they be applied to a single, consistent dataset as introduced here.

Accordingly, we compare and benchmark the eight distinct network architectures presented by the mentioned works and all models tested employ a neural network architecture containing the following components:

1. Convolutional layers trained to extract local features that are independent of position and orientation.
2. Fully-connected layers that learn classification rules based on the previously extracted features.

While the models all follow this general architecture, their exact implementation varies from paper to paper. Here, we test eight different models with architectures summarized in Table 1. The first three networks employ a straightforward structure containing stacked convolutional layers without any extra modifications (e.g. Batch-Normalization), followed by two fully connected layers. All three plain CNNs apply Maximum-pooling (maxpool) operations following the convolutional layers to reduce the dimensionality of the input scan. We further test more sophisticated networks that use batch-normalization between convolutional layers and activation functions (CNN BN), as well as a VGG-like architecture that stacks even more convolutional layers before the pooling operation (VGG). An important concept commonly found in neural networks applied to image data are residual blocks (Resnet) that use a large amount of convolutional layers to extract complex features, which is only possible by using “residual connections" that prevent the vanishing gradient problem in deep neural networks. Similarly, inception blocks (INC3 & INC6) combine different kernel sizes in convolutional blocks to account for different feature sizes.

We benchmark the different model implementations as they are presented in their respective papers, without any modifications. Each model is trained using an Adam classifier with default hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and a learning rate of $3e^{-4}$, with a mini-batch size of 128. Since different model architectures require more or less time to converge, we set a high number of training epochs (500) and use the Early-Stopping technique to halt training once the validation loss levels off. Additionally, we lower the learning rate by a factor of 0.5 if the validation loss does not improve for 10 consecutive epochs, while early stopping activates at 25 epochs without any improvement.

## Reproducibility

During our tests, we found that the network performance varied substantially depending on which random seed was used to initialize the training of each model. Therefore, we aim to benchmark not only the accuracy of each model, but also how much variance is present between different training iterations. The variation is further complicated by changes that can be introduced when training on GPU machines that utilize CUDA algorithms, which are not deterministic. Accordingly, the following policies are used to ensure reproducibility.

*Training in Docker containers*: Since the training of neural networks in a Python environment requires multiple libraries that are available in different versions, and we cannot confirm that every single combination produces identical results, we train our models in a Docker container. A Docker container is a small virtual machine that is built according to a recipe (image) and asserts platform-independent performance (Host machine can be Windows, MacOS or Linux). The container still has a few connections to the host machine (e.g. GPU drivers) that may be subject to change and could cause minor performance differences, but to our knowledge this is the most consistent option.

*Setting deterministic algorithm flags*: As mentioned before, TensorFlow (and other libraries like pyTorch) relies on CUDA algorithms that are not fully deterministic by default. Here, we set tensorflow to only use deterministic operations which hampers the computation speed but assures consistent results.

*Multiple model initializations*: We train each model 5 times to investigate performance differences between training runs. To account for random variations, we randomly set the seed at the start of each run, which determines how the model is initialized. We also shuffle the order of the training data between epochs. Using identical parameters such as the mini-batch size and random seed across models means that each network sees the exact same training samples per step and convergence only depends on initialization and the network's ability to learn meaningful features for the data.

## DATA AVAILABILITY

The data used to train the models, as well as the trained keras model (weights for the best-performing variant per model) are also available on figshare. The training process for each model was tracked on the Weights & Biases platform.

## CODE AVAILABILITY

All code used here can be found in the Github repository jschuetzke/synthetic-spectra-benchmark.

Received: 23 November 2022; Accepted: 23 May 2023;
Published online: 05 June 2023

## REFERENCES

1. Friedrich, W., Knipping, P. & Laue, M. Interferenzerscheinungen bei Röntgenstrahlen. *Ann. Phys.* **346**, 971–988 (1913).
2. Ernst, R., Bodenhausen, G. & Wokaun, A. *Principles of nuclear magnetic resonance in one and two dimensions* (Oxford Science Publications, 1987).
3. Smith, E. & Dent, G. *Modern Raman Spectroscopy: A Practical Approach* (John Wiley & Sons, 2019).
4. Belsky, A., Hellenbrandt, M., Karen, V. L. & Luksch, P. New developments in the inorganic crystal structure database (ICSD): accessibility in support of materials research and design. *Acta Crystallogr. B.* **58**, 364–369 (2002).
5. Lafuente, B., Downs, R. T., Yang, H. & Stone, N. The power of databases: The RRUFF project. In *Highlights in mineralogical crystallography*, 1–30 https://doi.org/10.1515/9783110417104 (De Gruyter (O), 2015).
6. Choudhary, K. et al. Recent advances and applications of deep learning methods in materials science. *npj Comput. Mater.* **8**, 59 (2022).
7. Szymanski, N. J. et al. Toward autonomous design and synthesis of novel inorganic materials. *Mater. Horiz.* **8**, 2169–2198 (2021).
8. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5**, 115–133 (1943).
9. Park, W. B. et al. Classification of crystal structure using a convolutional neural network. *IUCrJ* **4**, 486–494 (2017).
10. Wang, H. et al. Rapid identification of x-ray diffraction patterns based on very limited data by interpretable convolutional neural networks. *J. Chem. Inf. Model* **60**, 2004–2011 (2020).
11. Lee, J.-W. et al. A deep-learning technique for phase identification in multiphase inorganic compounds using synthetic xrd powder patterns. *Nat. Commun.* **11**, 86 (2020).
12. Schuetzke, J., Benedix, A., Mikut, R. & Reischl, M. Enhancing deep-learning training for phase identification in powder x-ray diffractograms. *IUCrJ* **8**, 408–420 (2021).
13. Szymanski, N. J. et al. Probabilistic deep learning approach to automate the interpretation of multi-phase diffraction spectra. *Chem. Mater.* **33**, 4204–4215 (2021).
14. Chen, D. et al. Review and prospect: Deep learning in nuclear magnetic resonance spectroscopy. *Chem. Eur. J.* **8**, 10391–10401 (2020).
15. Liu, J. et al. Deep convolutional neural networks for raman spectrum recognition: a unified solution. *Analyst* **142 21**, 4067–4074 (2017).
16. Ho, C.-S. et al. Rapid identification of pathogenic bacteria using raman spectroscopy and deep learning. *Nat. Commun.* **10**, 4927 (2019).
17. Kukula, K. et al. Rapid detection of bacteria using raman spectroscopy and deep learning. *Proc. IEEE Annual Computing and Communication Workshop and Conference* 796–799 (2021).
18. LeCun, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1**, 541–551 (1989).
19. Deng, J.et al. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 248–255 (2009).
20. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **60**, 84–90 (2017).
21. Kuhn, S. & Schlörer, N. E. Facilitating quality control for spectra assignments of small organic molecules: nmrshiftdb2 - a free in-house nmr database with integrated lims for academic service laboratories. *Magn. Reson. Chem.* **53**, 582–589 (2015).
22. Jain, A. et al. Commentary: The materials project: a materials genome approach to accelerating materials innovation. *APL Mater.* **1**, 012002 (2013).
23. Oviedo, F. et al. Fast and interpretable classification of small x-ray diffraction datasets using data augmentation and deep neural networks. *npj Comput. Mater.* **5**, 1–9 (2018).
24. Maffettone, P. M. et al. Crystallography companion agent for high-throughput materials discovery. *Nat. Comput. Sci.* **1**, 290–297 (2021).
25. Liang, Q., Dwaraknath, S. & Persson, K. A. High-throughput computation and evaluation of raman spectra. *Sci. Data* **6**, 135 (2019).
26. Bagno, A., Rastrelli, F. & Saielli, G. Predicting $^{13}$C nmr spectra by dft calculations. *J. Phys. Chem. A* **107**, 9964–9973 (2003).
27. Xu, Y. & Goodacre, R. On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *J. Anal. Test.* **2**, 249–262 (2018).
28. Zhang, R. et al. Transfer-learning-based raman spectra identification. *J. Raman Spectrosc.* **51**, 176–186 (2020).
29. Dong, H. et al. A deep convolutional neural network for real-time full profile analysis of big powder diffraction data. *npj Comput. Mater.* **7**, 1–9 (2021).
30. Davies, A. & Patiny, L. Nmrium browser-based nuclear magnetic resonance data processing. *Spectrosc. Eur.* **33**, 21–24 (2021).

## ACKNOWLEDGEMENTS

We want to thank the Karlsruhe House of Young Scientists (KHYS) for funding the scientific exchange that was essential for obtaining the results. N.J.S. was supported in part by the National Science Foundation Graduate Research Fellowship under grant #1752814. We also thank Gerbrand Ceder for the helpful discussion and invitation to UC Berkeley.

## AUTHOR CONTRIBUTIONS

All authors designed the research. J.S. and N.J.S. developed the approach to generate synthetic data. J.S. implemented and conducted the experiments. M.R. conceived and supervised the project. All participated in data analysis and writing the paper.

## FUNDING

Open Access funding enabled and organized by Projekt DEAL.

## COMPETING INTERESTS

The authors declare no competing interests.

## ADDITIONAL INFORMATION

**Supplementary information** The online version contains supplementary material available at https://doi.org/10.1038/s41524-023-01055-y.

**Correspondence** and requests for materials should be addressed to Markus Reischl.

**Reprints and permission information** is available at http://www.nature.com/reprints

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.