On the possible benefits of deep learning for spectral preprocessing

Runar Helin | Ulf Geir Indahl | Oliver Tomic | Kristian Hovde Liland

Faculty of Science and Technology, Norwegian University of Life Sciences, Ås, Norway

Correspondence
Runar Helin, Faculty of Science and Technology, Norwegian University of Life Sciences, Ås 1430, Norway.
Email: runarhel@nmbu.no

Abstract
Preprocessing is a mandatory step in most types of spectroscopy and spectrometry. The choice of preprocessing method depends on the data being analysed, and to get the preprocessing right, domain knowledge or trial and error is required. Given the recent success of deep learning-based methods in numerous applications and their ability to automatically detect patterns in data, we aimed at exploring the possibilities of using such methods for preprocessing. Our study considered a flexible but systematic investigation of spectroscopic preprocessing methods (classical and deep learning-based) combined with predictive modelling, including both traditional linear modelling and artificial neural network-based modelling. The main ambition of the present work was to assess if the advantages of deep learning-based methods in spectral preprocessing are sufficient to justify the additional efforts in model set-up and training and the possible losses of interpretability and transparency. With the use of data from different vibrational spectroscopy techniques, we demonstrated that deep learning-based preprocessing successfully increased the predictive performance of our models but that classical preprocessing still is a good alternative or even the best one in some cases. A significant increase in effort was required when using deep learning-based preprocessing together with linear model prediction. Compared with classical preprocessing techniques, deep learning-based preprocessing decreased the transparency and showed only modest improvements of the prediction performance of linear models. Our conclusion is that deep learning-based preprocessing is best suited when integrated in neural network predictions.

KEYWORDS
artificial neural networks, deep learning, model validation, preprocessing, spectroscopy

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.
© 2021 The Authors. Journal of Chemometrics published by John Wiley & Sons Ltd.

Journal of Chemometrics. 2022;36:e3374.
https://doi.org/10.1002/cem.3374
Our goal with this study is to make objective comparisons of various strategies for preprocessing and prediction, here termed pipelines. In addition to rigorous evaluation of the predictive performance for the proposed pipelines, we will also discuss the following key moments:

- Time required for fitting models and predictions based on new samples.
- Effort required to set up and tune the pipelines.
- Transparency and complexity of the pipelines and interpretability of the models.
- Robustness with respect to outliers and new data points.

Spectroscopy is the study of the interactions between electromagnetic radiation and matter. More specifically, it is the study of absorbance, emission and reflection of light at different energy levels in different samples. Vibrational spectroscopy is the subfield where the emitted light is affected by molecular vibrations in the molecular structure, manifesting itself as peaks or overtones at various wavelengths/wavenumbers in the spectra. Such data, which includes Raman and infrared spectra, contain a lot of information about atoms and molecules in the samples. Being high-dimensional and possibly highly correlated, the data can be challenging to analyse and to interpret. Preprocessing of the data is often required for the removal of irrelevant variation in the data, such as phenomena caused by light scattering, differences in temperature or differences in humidity. Traditional analysis methods include principal component analysis (PCA), partial least squares (PLS) regression and support vector machines (SVMs). However, there is limited research on the use of deep learning (DL) models with spectroscopic data. We begin by giving a brief introduction to the field of DL.

1.1 Deep learning

The history of artificial neural networks (ANNs) and DL can be traced back to the first descriptions of artificial neurons called Threshold Logic Units proposed by Warren McCulloch and Walter Pitts in 1943 and Rosenblatt’s perceptron in 1958. Important milestones in training and designing network architectures include the back-propagation algorithm invented in 1986, and the advent of convolutional neural networks (CNNs) in 1989. Further development led to the artificial intelligence (AI) revolution in image-based classification and descriptions, starting with the AlexNet winning the ImageNet competition in 2012.

An ANN has an architecture based on a series of interconnected units (neurons) organised in layers. The interconnections are sets of parameters \( w \) (weights) representing the strength of the connections between specific units in different layers of the network. These weights are the model parameters to be updated through minimisation of some loss function \( L(y, \hat{y}) \), where \( y \) is the true response and \( \hat{y} \) is the response predicted by the model. Common loss functions include the mean squared error (MSE): \( \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \) for regression problems and categorical cross-entropy, also known as the Bernoulli log-likelihood loss: \( \sum_{i=1}^{n} -y_i \log(\hat{y}_i) \) for classification problems, where \( y_i \) is a one-hot (dummy) encoded class label vector and \( \hat{y}_i \) is the corresponding vector of class probabilities predicted by the model. The categorical cross-entropy error function is a common choice in multiclass logistic regression and serves as a measure of the classification error on a continuous scale. The process of tuning the weights is referred to as training of the model. Training of an ANN is an iterative process, where one full cycle through the available training data for updating the model parameters is referred to as an epoch.

In the classical ANNs, all the units of one layer are connected to all of the units in the subsequent layer. Such models are often referred to as Multilayer Perceptrons (MLPs), fully connected feed-forward networks or simply a set of dense layers if included as modules in a more complex network architecture.

A key part of the ANNs is the activation function, which modulates the output of the layers. The activation function is usually non-linear, which gives the network the capability of representing complex non-linear relationships between the input and output data. A popular activation function alternative is the rectified linear unit (ReLU) defined as
Different arrangements of the interneuron connections and the number of layers (depth of the network) yield different network architecture options. In this study, we consider both MLPs and CNNs, characterised by the use of convolutions representing the connections between layers.

Originally designed for solving 2D image recognition problems, the CNNs include a number of convolutional filters at each layer, each with its own set of model weights. A key property of CNNs is the sparse connectivity between the layers, meaning that the inputs to each node of one layer are restricted to the outputs from nodes in a local neighbourhood (defined by the filter size) of the preceding layer. In addition, each filter is shifted across the input signal (image or spectrum) to allow the same filter parameters to be used on different locations of the input signal. The latter is also known as parameter sharing. Figure 1 illustrates the differences between a fully connected and convolutional layer. The neighbourhood restriction and parameter sharing reduce the number of network parameters compared with a fully connected architecture and is therefore computationally more efficient when training the CNN model. The filter size defines the so-called the receptive field of the layer and enables the network to account for spatial relationships in the input signals. This property has been proven extremely useful in problems involving object detection in images, such as recognition of handwritten digits. For more details regarding CNNs, see Goodfellow.

The use of ANNs with spectroscopic data was explored by Næs et al with an emphasis on near infrared (NIR) applications. More recent studies that utilise CNNs include Liu et al, whose neural network achieved superior classification results of mineral species based on their Raman spectra compared with other popular shallow machine learning methods such as K-nearest neighbour (KNN), SVMs and random forests. Acquarelli et al proposed a simple CNN architecture that outperforms popular linear models in chemometrics on a collection of popular datasets. The authors of these papers have also demonstrated that their ANNs achieve good regression performance using spectroscopic data without the need of a separate preprocessing step. Furthermore, Cui et al achieved good performance using CNN on NIR calibration, and Malek et al have proposed a CNN that uses an optimisation method based on particle swarms as an alternative to back-propagation. In our study, we focus on the possibility of using DL for preprocessing and explore how to use these methods in combination with linear prediction modelling.
1.2 | Preprocessing

1.2.1 | Classical preprocessing

In spite of the vast number of spectral preprocessing methods proposed in the literature, we have chosen to restrict our attention to the Savitzky–Golay filters, polynomial baseline corrections and intensity correction as accounted for by the extended multiplicative signal correction (EMSC). These are all popular methods for removal of known artefacts from the data caused by phenomena such as instrumentation and light scattering. Our choices represent a selection of methods applicable for different types of spectra, each requiring possibly different preprocessing approaches. The choice of preprocessing obviously influences the subsequent prediction modelling subject to validation of its predictive performance.

Savitzky–Golay

Savitzky–Golay filtering aims at smoothing a signal without corrupting its information content. The method is based on a sliding window approach where a polynomial curve is fitted locally to the data. Furthermore, the method can be implemented efficiently as a convolution operation. In addition to choosing different polynomial degrees, one can approximate the derivatives of the signal, which is useful for many types of noisy spectral data, especially Fourier-transform infrared (FTIR). The Savitzky–Golay filter has become a standard preprocessing tool in spectroscopic analysis within a wide range of applications. For equally spaced data points, the values of the filters can be found analytically, and implementations exist in both commercial and non-commercial software packages (such as the EMSC package in R and the SciPy ecosystem of Python-based open-source software).

Extended multiplicative signal correction

The EMSC signal correction method is a popular choice in vibrational spectroscopy. It is used for correction of global intensity differences and baselines in the spectra. The EMSC extends the multiple scatter correction (MSC) method and incorporates the possibility of also eliminating polynomial baseline trends in addition to the constant baselines handled by the ordinary MSC.

The EMSC considers a signal as represented by a constant term \( a \) together with a linear combination of a reference spectrum \( X_{\text{ref}} \) and additional terms corresponding to the polynomial trends of degree \( i = 1, \ldots, n \) \( (\nu^i) \) plus a residual term \( e \):

\[
X = a + X_{\text{ref}} \cdot b + \sum_{i=1}^{n} d_i \nu^i + e.
\]

The coefficients \( a \), \( b \) and \( d_i \) are estimated individually for each spectrum in the dataset, and the chemical variance is accounted for by the residual term, \( e \). The EMSC-corrected spectrum can then be expressed as follows:

\[
X_{\text{corr}} = \frac{X - a - \sum_{i=1}^{n} d_i \nu^i}{b} = X_{\text{ref}} + e.
\]  

Direct extensions of the EMSC method handle interferents, replicate variation, Mie scattering, multiple references and more. The EMSC method is useful for a range of different kinds of spectroscopic techniques, such as NIR, Raman and FTIR.

We also note the close relationship between the standard normal variate (SNV) transformation and the MSC. If the coefficients \( a \) and \( b \) in Equation (1) represent the mean and standard deviation of the spectrum \( X \), the expression corresponds to the associated SNV transformation. The main difference between the two is that MSC is a transformation based on a reference spectrum, where the mean spectrum is a common choice, whereas SNV is a transformation (i.e., centring and scaling) of each spectrum independently.

1.2.2 | DL-based preprocessing

DL models offer flexibility in design, can handle non-linearities and adapt to both known and unknown phenomena. Because of this, such models can sometimes be applied successfully, even without including all the prior knowledge
concerning the feature extraction procedures and required preprocessings to obtain successful applications of traditional statistical and chemometric models. Trainable preprocessing based on DL will inherit some of these traits.

In this paper, we present two preprocessing alternatives, both achieved by including trainable layers of a neural network. This approach combines the preprocessing and prediction steps of the data analysis problem into one unified model. The idea was first proposed by Dong et al. who introduced a model called Raman-CNN to classify blood samples based on their Raman spectra. Our first preprocessing alternative builds on their work, with two ANN layers being carefully designed for handling denoising and baseline correction, respectively. Our second alternative is a novel design of a neural network layer able to perform EMSC by evolving an appropriate candidate reference spectrum during the training process. We will refer to these alternatives as neural network denoising and baseline correction (NN-NoiseBase) and neural network EMSC (NN-EMSC), respectively.

These approaches have in common their trainable weights in the preprocessing layers. The outputs of the complete trained ANNs including the preprocessing layers can be considered directly as model predictions, where the preprocessing step and prediction model are combined in a single model. In addition, the outputs of the preprocessing layers can be considered as preprocessed (corrected) input data, also available for other choices of prediction modelling, including traditional linear models such as PLS. Furthermore, in our work, we consider two different classes of ANN architectures (MLP and CNN) attached after the preprocessing layers. The architectural details are given in the next section. In general, any choice of ANN is applicable. We have chosen to focus on these two architectures because they represent fairly general ANN architectures known from successful applications within many fields of analysis. Earlier studies using ANNs on spectroscopic data also include similar architectures.

The proposed NN-NoiseBase has two convolutional layers with added constraints on the weights of the convolution filters. An illustration of the configuration is included as Figure S1. The constraints require non-negative weights $w = [w_1, w_2, ..., w_k]^T$ (where $k$ is the filter size), which sum to 1:

$$w_i \geq 0, \sum_{i=1}^{k} w_i = 1, \text{ for } i = 1, 2, ..., k.$$  \hspace{1cm} (2)

These constraints ensure that the layers can actually evolve into meaningful filters performing denoising and baseline correction. Each of the two layers consists of a single filter. Furthermore, we deliberately omit the bias term in the convolution filters (to avoid shifting the outputs of the corresponding convolutional layers).

The denoising is obtained by a smoothing filter representing a local weighted average, and the filter size should be chosen experimentally just large enough to remove high-frequency noise from the spectra without affecting significant trends in the spectra. Thereafter, the baseline correction is achieved by using a wider smoothing kernel for capturing the main trends of the noise reduced spectra, which further is subtracted to obtain the baseline corrected data. With $h(\cdot)$ denoting the smoothing kernel, the corrected spectra can be expressed as follows:

$$X_{corrected} = X - X_{smoothed} = X - X * h(\cdot) = X * (I - h(\cdot)),$$  \hspace{1cm} (3)

where $I$ is the identity spectrum. The kernel $(I - h(\cdot))$ is then the baseline correction kernel. The required associated constraints are

$$w_i \leq I_i, \sum_{i=1}^{k} w_i = 0, \text{ for } i = 1, 2, ..., k,$$  \hspace{1cm} (4)

that is, the weights must sum to 0, and each weight must be smaller than the identity kernel $I$. See Dong et al. for more details about the derivation of this procedure. We expand on the Raman-CNN by considering NN feed-forward architectures that are not necessarily of the fully connected type proposed by the authors. Additionally, we apply the expanded Raman-CNN on various types of spectra. For this preprocessing approach, the sizes of the baseline correction and denoising filters are hyperparameters.

Our novel DL preprocessing technique, designed to perform EMSC, is implemented as an ANN layer that takes a raw spectrum as input and outputs the scatter-corrected spectrum. However, the reference spectrum is not predefined but considered as a vector of trainable weights, which makes the preprocessing step adaptive. Starting out with a
meaningful initialisation of the reference spectrum such as the mean spectrum, the reference spectrum weights are updated using the gradient descent algorithm as part of the loss minimisation during the network training process. The corresponding layer is implemented as a “Keras layer” in the terminology of the Tensorflow package. Similar to the classical EMSC applications, the choice of polynomial degree to be included in the correction is a hyperparameter to be chosen by the user.

1.3 Prediction models

To assess the utility of the DL-based preprocessing techniques, we compare the predictive performance of the preprocessed spectra using representatives of linear models and neural networks.

1.3.1 Linear modelling

There is a wide range of linear regression and classification methods routinely used with spectroscopic data. However, most of these achieve highly similar performance and robustness. We therefore choose a single proven representative, namely, PLS regression for our comparisons. We refer to Wold et al for historic roots and algorithmic details.

With PLSR, the input data are sequentially transformed into a subspace representation guided by the response(s), resulting in a lower-dimensional representation appropriate for prediction and interpretation. In contrast to PCA, the PLSR method is taking into account the response information available in regression and classification problems when determining the subspace representation. Using a dummy representation of categorical responses, the PLS methodology is also appropriate for classification purposes, then becoming PLS discriminant analysis (PLS-DA).

1.3.2 DL modelling

The choice of DL model architecture is often a challenging task. The number of layers and number of nodes per layer are in general problem dependent, based on experience and often found by trial and error. In our work, we based our choice of architecture on literature reviews, simple structure and testing on preliminary experiments. To be able to distinguish effects of the preprocessing techniques, the network architecture was kept fixed across all the experiments. However, in practical applications, architectural choices can be included in the tuning process for further optimisation.

Two different ANN architectures were considered in this work with an illustration of these architectures found in Figure S2 where more details are given.

The proposed architecture (A) is an MLP network with three hidden layers containing 128, 256 and 128 nodes, respectively. This is similar to the architecture used in Dong et al but with one additional hidden layer. The proposed architecture (B) is convolution based and consists of one hidden layer with 8 convolution filters of size (9 × 1) and one hidden dense layer with 32 nodes. CNNs used for computer vision problems are useful due to their ability to capture certain translational invariant features in the input images. In spectroscopy, the shapes and magnitudes of the spectra are of interest and not the spatial invariance. This suggests that not many convolutional layers are needed. In fact, practical experience shows that the added convolutional layers increase convergence speed but do not improve the prediction ability. For architecture (B), we also included batch normalisation of the outputs from the convolutional layer. Batch normalisation affects the propagation of the gradient during the training process and often results in faster convergence of the loss function minimisation. During our experimentation, we experienced that standardised (autoscaled) data were needed as input to the neural networks to achieve efficient convergence. Autoscaling makes the features homogeneous, and it ensures that each feature has equal influence on the gradient update and that the network weights and features have similar magnitude. In order to compare our DL-based preprocessing techniques with the classical ones, we used the raw data (without autoscaling) as input to the preprocessing layers. To obtain faster convergence, we included a batch normalisation layer after the preprocessing layers for both the NN-EMSC and NN-NoiseBase, acting as an adaptive scaling of the preprocessing stage.

Both the MLP and CNN architectures use ReLU activation functions for transforming the outputs from each intermediate layer. For the outputs of the final layer (the output node[s]), we used linear and softmax functions as activation functions for the regression and classification problems, respectively.
2 | DATASETS

In the present study, we have focused on two different datasets. The first dataset contains FTIR spectra of food by-product hydrolysates collected from a controlled experiment for the purpose of determining protein size distributions. The noise contained in these spectra is mitigated through the experimental set-up. Although such datasets usually are of high quality, they are often expensive to produce, meaning that the number of samples is often limited. It should be noted that ANN models often have difficulties in obtaining good generalisation when the sample size is too small.

The other dataset contains NIR spectra from a hyperspectral image. Each pixel of the image corresponds to a spectrum and is considered as a separate sample. Compared with datasets obtained from controlled experiments, such data may be more affected by noise but are generally cheaper to collect. Such pixel-based data contain a lot more data points and are more likely to be suitable for ANN modelling.

2.1 | The FTIR spectra

The first dataset represents a regression problem where the predictors are FTIR spectra of protein hydrolysates. The hydrolysates are made from various by-products from the food industry through enzymatic protein hydrolysis using different enzymes. There are 28 different by-product/enzyme combinations in total. Figure 2 shows some sample spectra. The response to be modelled is the corresponding (continuous) average molecular weight (AMW) measured by size exclusion chromatography. A detailed description can be found in Kristoffersen et al.\(^3\) In total, there are 885 spectra obtained from different time steps of the hydrolysis process. Additionally, the sampling of some by-product and enzyme combinations has been repeated, resulting in 332 unique samples when grouping by by-product, enzyme and time step. The spectra contain 1712 spectral bands in the range from 4000 to 400 cm\(^{-1}\). In our analysis, we limited the spectral region to 3700–400 cm\(^{-1}\) as the region above 3700 was without signal. This dataset has been studied by Kristoffersen et al.\(^3\) using classical models. They used a hierarchical modelling approach with a canonical PLS (CPLS) + linear discriminant analysis (LDA) model for classification of by-product/enzyme combinations as the first layer and a set of PLSR regression models for prediction of AMW as the second layer. We will use their modelling approach as our benchmark. As noted by the authors, the samples measured on by-products of turkey are challenging to predict because they are known to contain a larger amount of longer peptides at the start of the hydrolysis process in comparison with the other measured samples in the experiments. It could be claimed that the turkey samples should have been hydrolysed differently to obtain a peptide fraction of similar quality to the samples of chicken, salmon and mackerel.

**FIGURE 2** Samples from the two datasets. (A) One spectrum for each enzyme-material class in the Fourier-transform infrared (FTIR) dataset before preprocessing. The spectral range depicted is the same as passed to the models in the analysis. (B) Remote-sensing dataset. Top row: a sample image of one spectral band (1153 nm) with the corresponding label mask. Bottom: sample of one spectrum from each of the 16 classes.
2.2 The AVIRIS remote sensing data

Our second dataset represents a classification problem containing remote sensing data acquired by the AVIRIS instrument, a hyperspectral image showing the reflectance of different types of vegetation and soil types over an area in the Salinas Valley, California (see Figure 2). The hyperspectral image has $512 \times 217$ pixels with 224 spectral bands in the range from 400 to 2500 nm. The spatial resolution of the images is 3.7 m per pixel. In total, there are 16 different classes of vegetation and soil types. As response, we used the pixel-wise annotated class membership, making this dataset a representative of classification problems. Considering each pixel as a sample spectrum, the amount of data should theoretically suit DL models. For the model building, we use only a subset containing 8000 pixels and its corresponding spectra, in order to keep the computational cost lower. We also kept the relative sizes of the 16 classes fixed, meaning that the subset contained the same imbalance of the classes as the original image, having class sizes ranging from 1.7% to 20.85% of the pixels.

3 METHODS

In our study, we considered three families of preprocessing alternatives combined with predictive modelling, as illustrated in Figure 3:

1. The classical preprocessing alternative by EMSC and/or Savitzky–Golay filtering followed by training either a linear model (PLS or PLS-DA) (1), a fully connected feed-forward neural network (MLP) (2) or a CNN (3).
2. The NN-EMSC alternative including EMSC with an adaptive reference spectrum found during the training process of either an MLP (5) or a CNN (7). Alternatively, each of the resulting preprocessing parts obtained from the two trained neural models is used as filters before training a PLS-model (4, 6).

*FIGURE 3* Overview of the various pipelines considered. “SavGol” is short for Savitzky–Golay filtering. The two colors of “NN-NoiseBase” indicate that the method consists of two parts: denoising (green) and baseline correction (yellow)
3. The NN-NoiseBase alternative, which performs denoising and baseline correction preprocessing during the training process by either an MLP (9) or a CNN (11). Alternatively, each of the resulting preprocessing parts obtained from the two trained neural models is used as filters before training a PLS model (8, 10).

For the classical alternative, there are three model fitting pipelines. For each of the neural network-based preprocessing alternatives, there are four pipelines, because each alternative can be trained using either an MLP and a CNN as the main networks (two DL-based predictions) and a PLS model can be fitted for each alternative (two linear predictions).

Thus, our study considers a total of 11 pipelines, where all except of one include the training of an ANN. Additionally, we trained each of the three prediction models PLSR, MLP and CNN on the raw data to use as a benchmark for the preprocessing techniques. Each preprocessing method has its own set of hyperparameters that must be determined for each prediction model. To mitigate the complexity, we split our analysis into two phases: one phase concerning the selection of preprocessing hyperparameters and the other concerning the model selection, using the optimal parameters for each pipeline found in the first phase. Because the aim of the preprocessing hyperparameter search was to get a sense of which parameter and model combinations perform well, the number of epochs to train the ANNs was limited to 500 in this phase to make the search computationally feasible. However, in the model selection phase, the ANNs were allowed to train for up to 5000 epochs.

3.1 Validation

As a metric for evaluation, we used the root mean squared error (RMSE) for the regression problem and classification accuracy for the classification problem. Despite the imbalance in the data of the classification task, we found by inspection of the prediction accuracies of each class that the accuracy metric did not pose problems; that is, prediction accuracies of the large classes were not prioritised at the expense of the small classes. Figure 4 sketches the data splits used in the different phases of the analysis. All the segments were stratified to keep the original class balances. To validate the results, we used 75% of the data for the parameter selection phase (blue colour), leaving the final 25% as a test set for the model selection phase (orange colour). Furthermore, in the parameter selection phase, 2/3 of the data (Train data 1) was used in a threefold cross-validation to estimate the optimal number of PLS components and neural network epochs. The final 1/3 of the training data (validation data) was used to compute the prediction errors. When computing the prediction errors, all the samples used in the cross-validation (Train data 1) were used to fit the models for prediction.

In the model selection phase, all the samples from the parameter selection phase (Train data 1 + validation data) were used in a sevenfold cross-validation to estimate the optimal number of PLS components and neural network epochs. Similar to the parameter selection phase, all the samples from the cross-validation were reused to fit new models when computing the prediction errors. The prediction errors for each pipeline were computed on the 25% of the data not previously used (test data). The evaluation of the pipelines was based on these prediction errors.

Using a k-fold cross-validation is not the most common validation method of ANNs, where usually a single split is used as validation set. Most applications using DL models have plenty of data available; thus, a single validation split is often sufficient to accurately validate the models. However, we observed that the validation score was highly dependent on the split due to the number of samples and classes in our datasets. The cross-validation approach gave more stable and representative evaluations but came at the cost of training k neural networks instead of just one. In practice, this extra computational cost did not pose a problem for our analysis, because the total number of samples in each of the k neural networks was relatively small and training times correspondingly shorter.

**FIGURE 4** Data splitting scheme for the two analysis phases. “Training data 1” and “Validation data” are used for parameter selection. “Training data 2” and “Test data” are used for model selection. Note that “Train data 2” = “Train data 1” \cup “Validation data”
Special care had to be taken when validating the pipelines involving neural network-based preprocessing with a component-based linear prediction model such as PLS. In these pipelines, the predictive performance of the PLS model had to be evaluated for a selection of epochs of model training in order to allow the PLS model to determine the optimal number of epochs for training the preprocessors. In order to reduce the computational cost, we chose to evaluate the PLS models every 50 epochs in the parameter selection phase and every 25 epochs in the model selection phase. The resulting evaluation amounted to matrices containing the predictive performance for different number of epoch and PLS component combinations (Figure S3). From this matrix, the optimal number of epochs and components could be chosen based on the global optimum or by some other procedure.

3.1.1 | Parameter selection phase

The parameter selection phase was included as a step to reduce the complexity of the full model search. In this phase, training of all neural network models was limited to 500 epochs each, in order to finish the parameter search in feasible time. This did not guarantee convergence of every model, but experimentation showed that the number of epochs was sufficient to see the main trends and differences. For each preprocessing method, we performed a grid search over a range of selected hyperparameters as shown in Table 1.

For the NN-NoiseBase method, the sizes of the two preprocessing filters are hyperparameters. Different spectral data contain peaks with different widths. We adjusted the range of possible baseline correction filter sizes for each dataset, to allow the filter to cover the width at the base of the peaks.

An ANN is a stochastic model with a lot of randomness. The initial network weights are randomly drawn from some probability distribution, and the network training is stochastic because it uses randomly selected subsets of the data during the weight updates. Also, the inclusion of dropout layers in the network adds more randomness. The stochastic nature is useful in that it helps the model avoid local minima during optimisation and to find model weights that on average work well for the problem. However, the randomness causes different runs of the same model, using the same data set, to yield different results. This is usually more pronounced for models with small datasets. To mitigate the effects of this randomness, we computed the prediction error for each choice of hyperparameters several times, using different weight initialisations of the neural networks. During the parameter selection phase, we chose to compute the prediction error three times. The average prediction error across the three repetitions was used to determine the best hyperparameters for each pipeline. Optimally, the ANN models used during the cross-validation should also be trained with additional weight initialisations, but given the size of our grid search, it was too computationally expensive to perform.

3.1.2 | Model selection phase

The main assessment of the different preprocessing techniques, using either PLS or an ANN for prediction, was based on the results from the model selection phase. In this phase, we used the optimal hyperparameters found for each of
the pipelines during the parameter selection phase and validated the preprocessing techniques on more data. Additionally, we allowed the ANNs to train for more epochs. A sevenfold cross-validation scheme was used to find the optimal number of PLS components and the number of epochs for each pipeline. Initially, we computed the prediction errors three times for each pipeline, varying the random initiation of weights, similar to what was done in the parameter selection phase. However, the variation between each run was found to be larger than anticipated, especially for the FTIR dataset. Therefore, the prediction errors were computed 30 times, in order to get a more accurate estimation of the performances. The repeated computations of the prediction errors were computationally feasible because there were only 11 prediction models in this phase (one model per pipeline), compared with the parameter selection phase.

4 | RESULTS

Assessment of the pipelines involving DL-based models is challenging due to the stochastic nature of the ANN training process. Because finding the globally best hyperparameters was not the focus in this study but rather to obtain sufficiently good models for fair comparison, we automated the process and did not assess all of the model variability in the parameter search phase. However, it was possible to discern some patterns across the hyperparameters for the different pipelines. We therefore start by presenting these observations.

We experienced that the model predictions for the FTIR dataset depended heavily on both the data size and the samples of each subset. In general, we observed that the cross-validation errors were larger than the test-set prediction errors for all pipelines, with a greater difference between the two in the parameter selection phase compared with the model selection phase. As stated earlier, the choice of the optimal preprocessing hyperparameters for each pipeline was based on the prediction errors on the validation data (see Figure 4) and not the cross-validation error.

4.1 | The preprocessing hyperparameters

In accordance with our expectations, we observed that the same set of hyperparameters was not equally good for modelling of the two datasets in our study.

Starting with the FTIR dataset, the best parameter choices for the PLS model did not include EMSC. On the other hand, the EMSC technique with second-order polynomial correction was preferred by the PLS prediction model on the remote-sensing dataset. The Savitzky–Golay filtering without estimation of derivatives was favoured by PLS for both datasets. For the MLP and CNN prediction models, the best predictions were achieved when not including the EMSC technique. Different from the best PLS model, however, was that the inclusion of first derivative estimation in the Savitzky–Golay filtering gave the best predictions for both MLP and CNN modelling with the FTIR dataset. The same Savitzky–Golay filter parameters were also found for the best models based on the remote-sensing dataset; however, CNN modelling also achieved almost the same predictive performance without including the derivative estimation.

For NN-EMSC preprocessing, the critical hyperparameter is the degree of the polynomial trend in the correction. It was found that the better choice when using PLS as the prediction model was degree 0 for the FTIR dataset (resulting in MSC) and degree 1 for the remote-sensing dataset. Interestingly, both the MLP and CNN models gave better predictions with a second-order polynomial trend for both datasets.

For the NN-NoiseBase alternative, we observed a clear difference in the best choices of the hyperparameters for the different types of prediction models. The prediction results on the validation data for the different parameter choices are summarised in Figure 5. The pipelines using a PLS prediction model showed better performances when large filters were used, whereas the opposite was true for the pipelines using an ANN prediction model. The figure also illustrates that the ANN prediction pipelines contained a larger variability in model performances compared with the PLS pipelines.

4.2 | Pipeline comparison

The numerical results from the model selection phase are summarised in Table 2. The prediction performance, measured by the root mean squared error of predictions (RMSEP) and accuracy of predictions (AccP) (proportion correctly classified), is also shown in Figure 6. Notice that for both datasets, the best predictions were achieved by the pipelines with ANNs as prediction model.
Taking a closer look at the FTIR results, we see that the best predictions were obtained using the NN-NoiseBase preprocessing technique in combination with an MLP. The NN-EMSC preprocessing technique and SG-EMSC (classical) preprocessing technique both achieved predictions close to the NN-NoiseBase technique when combined with a CNN. Somewhat surprisingly, the CNN model gave very good predictions for the raw data compared with the other pipelines. The poorest predictions were obtained by the pipeline using the NN-EMSC in combination with

**FIGURE 5** The prediction results on the validation set during the parameter selection phase of the pipelines using the NN-NoiseBase preprocessing technique for different sizes of the denoising and baseline-correction filters. Left column: Fourier-transform infrared (FTIR) dataset. Right column: remote-sensing dataset. Each line shows the mean score with standard error as bars.

**TABLE 2** Prediction performance of the model selection phase

| Preproc. method | Preproc. engine | Prediction model | RMSE-CV | RMSEP     | AccCV   | AccP            |
|-----------------|---------------|-----------------|--------|-----------|--------|-----------------|
| Raw             | MLP           | Raw MLP         | 512    | 465.0 ± 3.01 | 0.910  | 0.91 ± 0.0016  |
| Raw             | CNN           | Raw CNN         | 426    | 353.0 ± 7.0  | 0.925  | 0.926 ± 0.0012 |
| Raw             | PLS           | Raw-PLS         | 457    | 426.0 ± 0.0  | 0.913  | 0.914 ± 0.0    |
| SG-EMSC         | MLP           | SG-EMSC MLP     | 439    | 357.0 ± 2.57 | 0.949  | 0.948 ± 0.0007 |
| SG-EMSC         | CNN           | SG-EMSC CNN     | 401    | 323.0 ± 3.96 | 0.950  | 0.948 ± 0.001  |
| SG-EMSC         | PLS           | SG-EMSC PLS     | 457    | 415.0 ± 0.0  | 0.917  | 0.918 ± 0.0    |
| NN-EMSC         | MLP           | NN-EMSC MLP     | 543    | 514.0 ± 5.92 | 0.929  | 0.929 ± 0.001  |
| NN-EMSC         | CNN           | NN-EMSC CNN     | 427    | 319.0 ± 6.93 | 0.928  | 0.922 ± 0.0017 |
| NN-EMSC         | PLS           | NN-EMSC PLS     | 442    | 421.0 ± 0.35 | 0.915  | 0.915 ± 0.0001 |
| NN-EMSC MLP PLS | CNN           | NN-EMSC MLP PLS | 420    | 406.0 ± 1.57 | 0.916  | 0.916 ± 0.0001 |
| NN-NoiseBase    | MLP           | NN-NoiseBase MLP| 427    | 316.0 ± 4.34 | 0.939  | 0.938 ± 0.0029 |
| NN-NoiseBase    | CNN           | NN-NoiseBase CNN| 425    | 336.0 ± 4.2  | 0.934  | 0.926 ± 0.0017 |
| NN-NoiseBase    | PLS           | NN-NoiseBase PLS| 450    | 413.0 ± 1.1  | 0.916  | 0.912 ± 0.0002 |
| NN-NoiseBase    | CNN           | NN-NoiseBase CNN| 455    | 425.0 ± 1.57 | 0.916  | 0.914 ± 0.0004 |

Note: The RMSECV column is the average prediction error for the sevenfold cross-validation. The RMSEP column is the mean prediction error of 30 repeated trials. The AccCV column is the prediction accuracy for the sevenfold cross-validation. The AccP column is the mean prediction accuracy of 30 repeated trials, including the standard error. The bold font indicates the pipelines with the lowest RMSEP and the highest AccP.

Abbreviations: AccP, accuracy of predictions; CNN, convolutional neural network; MLP, multilayer perceptron; NN-EMSC, neural network extended multiplicative signal correction; NN-NoiseBase, neural network denoising and baseline correction; PLS, partial least squares; RMSE-CV, root mean squared error cross-validation; RMSEP, root mean squared error of predictions; SG, Savitzky–Golay.

Taking a closer look at the FTIR results, we see that the best predictions were obtained using the NN-NoiseBase preprocessing technique in combination with an MLP. The NN-EMSC preprocessing technique and SG-EMSC (classical) preprocessing technique both achieved predictions close to the NN-NoiseBase technique when combined with a CNN. Somewhat surprisingly, the CNN model gave very good predictions for the raw data compared with the other pipelines. The poorest predictions were obtained by the pipeline using the NN-EMSC in combination with
the MLP model. The general impression is that the pipelines including neural networks with convolutions, either in the form of a CNN prediction model or in the form of filters in the NN-NoiseBase preprocessing stage, performed better. The predictions of the pipelines using PLS models did not vary as much as those using ANNs. All the preprocessing techniques resulted in better predictions by the PLS model compared with predictions on the raw data. The best predictions were achieved using the novel NN-EMSC technique, with the NN-EMSC filter fitted using the CNN model (Pipeline 6). Note that by including DL-based preprocessing filters (NN-EMSC or NN-NoiseBase) in the pipelines with PLS models, we introduce variance as indicated by standard errors of AccP. This needs to be taken into account in the comparisons.

For the classifications with the remote-sensing dataset, the best predictions were achieved using the classical preprocessing techniques in combination with either an MLP or a CNN model. The classical preprocessing was also the best choice for PLS modelling. Comparing the PLS pipelines, the poorest predictions were achieved by the pipeline using the NN-NoiseBase technique with MLP as the preprocessing engine. Compared with the predictions for the raw data, the DL-based preprocessing techniques improved the PLS predictions only slightly on this dataset. Similarly as for the FTIR dataset, the CNN model predicted fairly well also on the raw data. A difference from the FTIR dataset is that the NN-EMSC technique worked better with the MLP model than with the CNN model. This illustrates that the choice of ANN architecture is critical in combination with the DL-based preprocessing techniques proposed in this paper.

### 4.3 Time usage

Table 3 shows the time usage of each prediction model in our experiment for both datasets. The table gives the time (in seconds) used to fit the model to the training set and make predictions on the test set. The timing was performed on the model selection phase.

From the table, it is clear that the PLS models had far less run time compared with the ANNs. For the remote-sensing dataset (the largest one), the slowest model was a CNN using the NN-EMSC technique, which used approximately 14 min to complete the 5000 training epochs on our computer. The largest time sink in our experiment was caused by the validation step. During this step, the pipelines containing a DL-based preprocessing technique followed by PLS prediction were the slowest. This is because our experimental setup required repeated fitting of the PLS model during training of the DL-based preprocessing layer. With the PLS evaluations for every 50 epochs in our set-up, the total number of model evaluations was 200 during our 5000 epochs of training for each pipeline. Despite the quick evaluation of the PLS models, these evaluation slowed down the executions due to the additional cross validation steps. During the model selection phase with three repeated evaluations on the test set, all the pipelines using classical preprocessing (Pipelines 1–3) had a combined run time of 26 min for the FTIR dataset and 2 h for the remote-sensing dataset.
dataset. The pipelines using the NN-EMSC technique (4–7) had a combined runtime of 2 and 15.5 h, whereas the pipelines using the NN-NoiseBase technique (8–11) had a combined runtime of 1 h and 50 min and 14 h and 40 min, respectively, for the two datasets. Rerunning the test set predictions with 30 repetitions required approximately 5 h for the FTIR dataset and 18.5 h for the remote-sensing dataset.

| Classical alternatives | FTIR (in s) | Remote-sensing (in s) |
|------------------------|-------------|-----------------------|
| PLS/PLS-DA             | 1.0         | 15.4                  |
| SavGol                 | +0.011      | +0.022                |
| EMSC                   | +0.024      | +0.027                |

| DL alternatives        | FTIR (in s) | Remote-sensing (in s) |
|------------------------|-------------|-----------------------|
| MLP                    | 107.9       | 352.9                 |
| CNN                    | 121.4       | 467.2                 |
| NN-EMSC (MLP)          | +40.7       | +257.4                |
| NN-EMSC (CNN)          | +49.2       | +250.3                |
| NN-NoiseBase (MLP)     | +27.8       | +205.7                |
| NN-NoiseBase (CNN)     | +36.6       | +205.0                |

Note: The table shows the time for fitting the training data and predicting test data in the model selection phase. The PLS models used 70 and 120 components on the FTIR and remote-sensing datasets, respectively, whereas each ANN was trained for 5000 epochs. The ANN modelling was performed on an NVIDIA RTX8000 graphics card, whereas the PLS models were run on an AMD EPYC 7302 16-Core Processor.

Abbreviations: CNN, convolutional neural network; FTIR, Fourier-transform infrared; MLP, multilayer perceptron; NN-EMSC, neural network extended multiplicative signal correction; NN-NoiseBase, neural network denoising and baseline correction; PLS-DA, partial least squares discriminant analysis; RMSE-CV, root mean squared error cross-validation; RMSEP, root mean squared error of predictions.

5 | DISCUSSION

5.1 | DL-based preprocessing with ANN prediction models

In this work, we have explored how one can use DL-based techniques to perform spectral preprocessing. In the pipeline configurations containing both DL-based preprocessing and an ANN prediction model, the whole pipeline is a single seamless neural network but with added constraints on the first few layers. The constraints give us some idea about the role of these layers, but the whole model is mainly of the black box type. The ANNs using the DL-based preprocessing techniques resulted in better predictions compared with vanilla ANNs trained on raw data, especially regarding the FTIR dataset. Remarkably, the training of our proposed CNN architecture resulted in good prediction performance even when trained directly on the raw data without any preprocessing. When training the network on raw data, adequate preprocessing seems to be handled implicitly by the network. The success of ANN prediction on raw data is in accordance with the observation done by Liu et al.16 While we were able to improve the predictions of the ANNs by explicitly adding either classical or DL-based preprocessing, the additional efforts required to tune the DL-based preprocessing parameters may be considered superfluous. Another additional step required for the DL-based preprocessing is the tuning of hyperparameters. With little or no domain knowledge, the hyperparameter search may be time-consuming. Obviously, the choice of ANN architecture is also crucial to obtain a resulting model that predicts well. In our experiments, the use of the NN-EMSC technique in combination with an MLP turned out to be a poor alternative for the FTIR dataset. However, the same preprocessing technique combined with a CNN resulted in a very good model. For the remote-sensing dataset, the configurations using MLP were more favourable for both the NN-EMSC and NN-NoiseBase techniques. The choice of a good network architecture is a complicated matter, and by introducing further hyperparameters with the DL-based preprocessing techniques, the efforts and time required to set up the experiment will be rather extensive. Even on our small selection of datasets, we experienced that one specific network architecture is not effective for every pipeline configuration.
5.2 DL-based preprocessing with PLS prediction models

When including neural networks in the preprocessing step, we introduced additional model variance in the PLS predictions. This variance is caused by the random weight initialisation and the random batch selections (during the network training) of the ANNs used to train the DL-based preprocessing filters. This implies that the weights in the preprocessing filters most likely converge to different values (corresponding to local minima of the associated optimisation problem). A possible disadvantage with the approach using part of a trained neural network prior to a PLS model is that the prediction errors of the PLS model do not influence the updating of the weights in the preprocessing layer of the neural network. This was the reason why we needed to repeatedly fit PLS models for validation when using DL-based preprocessing in combination with PLS model predictions. The argument we make for the DL-based preprocessing is that the trained preprocessing layers will be adequate for any prediction model because, by design, they represent a kind of preprocessing such as denoising, baseline correction or EMSC. With good choices for the NN-architecture, we have demonstrated that this approach can in fact increase the predictive performance of the subsequent PLS model. However, the model selection- and validation process for this approach is much more time-consuming than for the traditional preprocessing methods combined with ordinary PLS modelling. A way to simplify the model selection and validation process could be based on some clever way of incorporating the prediction errors from the PLS model with the optimisation of the neural network weights.

5.3 ANN as a feature extractor for linear models

In a neural network, all the the layers except the final one are trained such that the network non-linearly transforms the input data for obtaining the best possible predictions from the final layer. In other words, one can think of the early layers of the network as useful feature extractors calculated from the input data. Besides supporting the predictions of the neural network itself, these features can be used as inputs for any prediction model. The idea of considering an ANN as a feature extractor for spectroscopic signal regression was explored by Malek et al.\textsuperscript{21} By discarding the final layer of a network after the training process was completed, they used the ANN model exclusively as a feature extractor, providing input data the prediction models used in their analysis.

In our work, we are doing something similar by discarding all parts of the trained neural network that are not corresponding to the trained preprocessing filters of the NN-EMSC and NN-NoiseBase techniques. As illustrated above, ANNs are able to implicitly perform adequate preprocessing. When discarding a subset of the preprocessors or feature extractors. It may well be that useful parts of the preprocessing actually occur after the dedicated preprocessing layers. Because the output of the preprocessing layers is not necessarily the best for linear models such as PLS, careful model validation is obviously extremely important. As discussed above, the preprocessing layers of the DL-based preprocessing techniques may provide more useful information for the ANN prediction models than for the PLS prediction models. This begs the question whether the neural network by itself produces features in the preprocessing layers that are useful for the PLS model or if some feedback from the PLS modelling part is needed. The problem would then be reduced to fitting only a single model, which would lead to a simpler validation process as well as possibly faster convergence and better concordance between the choices of the preprocessing weights and resulting model performance.

5.4 Interpretability

Another issue worth discussing concerning the DL-based preprocessing is the lack interpretability. The classical preprocessing techniques have been designed for the purpose of eliminating unwanted variation from specific sources with known characteristics. This part is omitted when introducing neural networks in the preprocessing procedure. Although our DL-based preprocessing techniques are mimicking classical methods, the neural network weights of the preprocessing filters are determined from the data by the complex relationships modelled by the chosen ANN architecture(s). Therefore, there is no precise way to conclude exactly how the DL-based preprocessing is correcting the data. The preprocessing layers may act differently depending on the choice of prediction model. Figure 5 supports this hypothesis. Compared with predictions on raw data, the NN-NoiseBase technique was successful both with the MLP as the prediction model and when used as a feature extractor for a PLS model. When
looking at the optimal hyperparameters of this preprocessing alternative (the filter sizes) for the MLP and PLS models, we see that the MLP gave better model predictions with short filters, whereas the PLS gave better model predictions with longer filters.

As pointed out in the Section 2, the FTIR dataset contains some samples measured on turkey, which are considerably harder to model than the other samples. Therefore, we also included the test set predictions for only the turkey samples in Figure 6. Note that the predictions of the turkey samples were much better for a CNN trained on the raw data compared with any of the preprocessing pipelines. A possible explanation for this phenomenon may be that the CNN model is leveraging raw-material information in the spectra when modelling the AMW, thus obtaining a more robust model. Except for the combination of NN-NoiseBase and MLP, the DL-based preprocessing was not capable of providing features to predict the turkey samples well. This indicates that an ANN model trained on the raw data may be more robust to new unseen data. It should also be noted that the raw data used to train the ANNs were autoscaled. As explained in Section 1.3.2, the autoscaling was a necessary preprocessing step for good convergence. The use of autoscaling is not common in spectroscopy where interpretation is important. However, in the setting of neural networks, we argue that it should not be avoided. In contrast to PLS, ANNs do not rely on a low-dimensional latent space to fit its model easily. Therefore, it is more beneficial to make all the features have similar distributions at the expense of losing the connection between features through autoscaling. Furthermore, when using ANNs, the possibility of interpretation is heavily reduced, and the use of autoscaling will not change that aspect. Despite the overall superior performance of the DL-based preprocessing, it is interesting that the classical SG-EMSC preprocessing technique seems to work better for the ANNs to predict the turkey samples. These remarks indicate some of the complexities and challenges associated with ANN modelling. In order to get improved predictive performance on the FTIR dataset, Kristoffersen et al. performed a two-level modelling approach (classify enzyme/material combination, then predict AMW from local model). For comparison, we repeated their modelling approach on the same train-test split as described in our paper. We applied preprocessing using Savitzky–Golay filtering followed by EMSC and used the spectral range between 1800 and 700 cm\(^{-1}\) as done in their paper. Using a single PLS model fitted on the training data (75% of the total data), the RMSEP for the test data was 454. Their two-level approach yielded an RMSEP of 280, which beats all our pipelines, while still having a linear and transparent model. This demonstrates that clever use of classical models can beat DL models on this particular dataset. Our best pipeline using DL prediction achieved an RMSEP of 316, which is also very good but required a considerable effort to achieve. A slight improvement might be possible by further tuning the ANN architecture and model parameters but then further increasing the effort. This shows that the DL models can do a great job working with spectral data, even with a limited amount of data. However, classical modelling schemes should not be underestimated.

5.5 | Neural networks and small datasets

When working with small datasets, the model validation process becomes very important in order to prevent overfitting and poor predictions on new unseen data. The FTIR dataset contains more features than samples and in the context of this experiment is considered a small dataset. We believe that the observed large variation in the prediction performance of the ANNs for this dataset is partly explained by the relatively low number of samples. It is also possible that some variation can be explained by the fact that the objective of the FTIR dataset is regression in contrast to the classification problem for the remote-sensing dataset. During our experiment, we observed two kinds of variations in the prediction performance. One was the variation when training from scratch, which means that the ANN models are sensitive to the weight initialisation and gradient updates. The other variance was seen in the difference between RMSECV and RMSEP for all the pipelines, which suggests a sensitivity in the splitting of the dataset. In addition to the data size, the variations can be explained by the fact that the dataset is heterogeneous in the sense that it has many subgroups, which increases the complexity of the underlying subspace. This fact was demonstrated by a previous analysis of the dataset. Despite the difficulties regarding this dataset, our experiment shows that the ANNs performed well. However, special care had to be taken in the validation process to confidently arrive on such a conclusion. With small datasets, we suggest that the prediction performance of ANNs should be reported as an average of training using different weight initialisations in order to assess some variability. This will make the results more convincing and add confidence in the ANN model.
5.6 | Time usage

In the Section 4, we reported the time usage of each prediction model. None of the datasets we used in our experiment can be considered as particularly large, and our network architectures are relatively shallow compared with the common architectures used in many ANN applications. Therefore, the training of a single neural network was not particularly time-consuming. Although the runtimes of the ANNs were considerably larger compared with the PLS models, they were still comfortably within the range of practical use on a personal computer. As explained in the Section 4, the validation of each pipeline was the most time-consuming part of the modelling process. Careful validation of the PLS models was important in order to make an accurate assessment of the predictive performance of each pipeline, but the additional cost required makes this approach rather unattractive. Based on the time usage alone, it is evident that our DL-based preprocessing alternatives are more suitable as integrated parts of a neural network and not for subsequent linear predictions.

5.7 | Efforts required

The classical preprocessing + PLS modelling pipeline was straightforward and did not require much efforts to set up. All elements of this pipeline are readily available in both commercial and open software packages and are straightforward to combine. Without much risk of making serious mistakes, several of the preprocessing parameters can be assumed as fixed in advance based on the type of spectra to be analysed. And from a user’s perspective, the comforting effect of obtaining deterministic results (same result if run again) should not be underestimated.

We had to put a lot of efforts into setting up and tuning the different pipelines involving the ANN modelling. A large part of these efforts went into testing the alternative architectural choices of the MLP and CNN models. Different depths of the networks, choice of activation functions, choice of the number of convolution filters and regularisation alternatives such as dropout were tested. Over time, we accumulated some confidence in what choices that were likely to work well on our datasets. When comparing the network architectures, we had to make sure these networks contained sufficient complexities to account for the structures of the data without being too prone to overfitting.

Regarding the ANNs, hyperparameters such as the learning rate and the batch size had to be determined. These hyperparameters generally affect both the speed of the convergence and the value of the converged loss function. Additionally, we observed that the convergence rate (in number of epochs) depended on the choice of preprocessing method, with the NN-EMSC method being the slowest alternative.

For the pipelines including both the DL-based preprocessing and ANN prediction modelling, an additional consideration about the number of epochs between the PLS predictions was needed to obtain a fair assessment of prediction power.

Ideally, the hyperparameters related to the ANN modelling should be included in the grid search together with the preprocessing parameters in case of interaction effects between the two sets of parameters. However, this parameter search alternative quickly became overwhelming when combining the different choices of ANN architectures, the training parameters (like learning rate and epochs) and the preprocessing parameters. Therefore, we decided to use two fixed ANN architectures based on some trial and error to do prediction modelling for our datasets and split the search for best preprocessing parameters and comparison of pipelines into two separate experiments.

Tuning of the pipelines was relatively slow because this required training of multiple ANN models. Another complication was that the ANNs were very sensitive to the weight initialisation when trained with the FTIR data. Because of this, we had to tune the pipelines based on multiple runs and not just a single one. From the knowledge we have gained through this work, we expect the set-up for a new experiment to be easier and faster; however, tuning the pipelines will still require considerable extra efforts compared with the classical pipeline including some preprocessing alternative followed by PLS modelling.

6 | CONCLUSION

In this study, the predictions obtained by the ANN models were generally better than the predictions obtained by the PLS models. This indicates the usefulness of such models for prediction modelling based on vibrational spectroscopic data. However, a careful assessment of the model variance is required before definite conclusions can be made. In our
study, the best prediction results for the FTIR dataset were achieved by training the NN-NoiseBase preprocessor combined with an MLP. The best prediction results on the remote-sensing dataset were obtained using the classical SG-EMSC preprocessing alternative, which provided the inputs for training an MLP model. Across all the preprocessing alternatives, the MLP prediction model alternative resulted in both the best and worst models, emphasising the importance of choosing a proper preprocessing alternative. On the other hand, the prediction results obtained by using CNN models did not vary as much across the different preprocessing alternatives and were outperformed by the best MLP models with less than one standard error. Our results therefore indicate that convolutional-based neural networks may be the more robust alternative, which also have the ability to capture an implicit preprocessing of the spectra better than the MLP models. We have also demonstrated that some DL-based preprocessing alternatives are capable of improving the predictive performance of PLS modelling when compared with the classical pipelines. However, these improvements come at the cost of longer training time and a larger effort in setting up the ANN modelling pipeline supplying the preprocessing filters. For quick analyses of spectroscopic data, the reliance on Beer–Lambert’s law and classical methods are still relevant, but for models to be used over time, small improvements in predictions may be worth the extra effort of DL-based preprocessing.

CONFLICT OF INTEREST
The authors declare that they have no conflicts of interest.

PEER REVIEW
The peer review history for this article is available at https://publons.com/publon/10.1002/cem.3374.

ORCID
Runar Helin https://orcid.org/0000-0001-7455-3108
Ulf Geir Indahl https://orcid.org/0000-0002-3236-463X
Oliver Tomic https://orcid.org/0000-0003-1595-9962
Kristian Hovde Liland https://orcid.org/0000-0001-6468-9423

REFERENCES
1. Rinnan Å. Pre-processing in vibrational spectroscopy-when. Anal Methods. 2014;6(18):7124-7129.
2. Pearson K. On lines and planes of closest fit to points in space. Philos Mag. 1901;2:559-572.
3. Wold S, Martens H, Wold H. The multivariate calibration problem in chemistry solved by the PLS method. In: Kågström B, Ruhe A, eds. Matrix Pencils. Lecture Notes in Mathematics. Vol 973. Havsbad, Sweden: Springer; 1983:286-293. https://doi.org/10.1007/BFb0062108
4. Wold S, Ruhe A, Wold H, Dunn Iii WJ. The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. SIAM J Sci Stat Comput. 1984;5:735-743.
5. Cortes C, Vapnik V. Support-Vector Networks. Mach Learn. 1995;20(3):273-297.
6. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychol Rev. 1958;65:386-408.
7. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986a;323:533-536.
8. Rumelhart DE, Hinton GE, Williams RJ. Learning Internal Representations By Error Propagation. Technical report, San Diego, Institute for Cognitive Science; 1986b.
9. Denker JS, Gardner WR, Graf HP et al. Neural network recognizer for hand-written zip code digits. In: Advances in Neural Information Processing Systems (NIPS 1989). Denver, CO, USA; 1989.
10. LeCun Y, Boser B, Denker JS, et al. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1989;1:541-551.
11. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. Lake Tahoe, NV, USA; 2012.
12. Bishop CM. Pattern Recognition and Machine Learning. New York, NY: Springer; 2006. https://doi.org/10.1007/978-0-387-45528-0
13. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278-2324.
14. Goodfellow I. Deep Learning. Boston, MA, USA: MIT Press; 2016. https://www.deeplearningbook.org
15. Næs T, Kvaal K, Isaksson T, Miller C. Artificial neural networks in multivariate calibration. J Near Infrared Spectrosc. 1993;1:1.
16. Liu J, Osadchy M, Ashton L, Foster M, Solomon CJ, Gibson SJ. Deep convolutional neural networks for Raman spectrum recognition: a unified solution. Anal. 2017;142:4067-4074.
17. Altman NS. An introduction to kernel and nearest-neighbor nonparametric regression. Am Stat. 1992;46(3):175-185.
18. Ho TK. The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell. 1998;20(8):832-844.
19. Acquarelli J, van Laarhoven T, Gerretzen T, Tran TN, Buydens LM, Marchiori E. Convolutional neural networks for vibrational spectroscopic data analysis. Anal Chim Acta. 2017;954:22-31.
20. Cui C, Tom Fearn.. Modern practical convolutional neural networks for multivariate regression: applications to NIR calibration Chemometrics and Intelligent Laboratory Systems. 2018;182:9-20.
21. Malek S, Melgani F, Bazi Y. One-dimensional convolutional neural networks for spectroscopic signal regression. J Chemom. 2018;32: e2977.
22. Martens H, Stark E. Extended multiplicative signal correction and spectral interference subtraction: new preprocessing methods for near infrared spectroscopy. J Pharm Biomed Anal. 1991;9:625-635.
23. Savitzky A, Golay MJ. Smoothing and differentiating of data by simplified least-squares procedures. Anal Chem. 1964;36:1627-1639.
24. Skogholt J, Liland KH, Indahl UG. EMSC: Extended Multiplicative Signal Correction. https://cran.r-project.org/package=EMSC; 2020.
25. Virtanen P, Gommers R, Oliphant TE, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020; 17:261-272.
26. Geladi P, MacDougall D, Martens H. Linearization and scatter-correction for near-infrared reflectance spectra of meat. Appl Spectrosc. 1985;39:491.
27. Martens H, Bruun SW, Adt I, Sockalingum GD, Kohler A. Pre-processing in biochemometrics: correction for path-length and temperature effects of water in FTIR bio-spectroscopy by EMSC. J Chemom. 2006;20:402-417.
28. Kohler A, Sulé-Suso J, Sockalingum GD, et al. Estimating and correcting Mie scattering in synchrotron-based microscopic Fourier transform infrared spectra by extended multiplicative signal correction. Appl Spectrosc. 2008;62:259-266.
29. Kohler A, Böcker U, Warringer J, et al. Reducing inter-replicate variation in fourier transform infrared spectroscopy by extended multiplicative signal correction. Appl Spectrosc. 2009;63:296-305.
30. Skogholt J, Liland KH, Indahl UG. Preprocessing of spectral data in the extended multiplicative signal correction framework using multiple reference spectra. J Raman Spectrosc. 2019;50:407-417.
31. Liland KH, Kohler A, Afseth NK. Model-based pre-processing in Raman spectroscopy of biological samples. J Raman Spectrosc. 2016;47: 643-650.
32. Helland IS, Næs T, Isaksson T. Related versions of the multiplicative scatter correction method for preprocessing spectroscopic data. Chemom Intell Lab Syst. 1995;29:233-241.
33. Dong J, Hong M, Xu Y, Zheng X. A practical convolutional neural network model for discriminating Raman spectra of human and animal blood. J Chemom. 2019;33:e3184.
34. Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-scale machine learning on heterogeneous systems; 2015. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf
35. Indahl UG. The geometry of PLS1 explained properly: 10 key notes on mathematical properties of and some alternative algorithmic approaches to PLS1 modelling. J Chemom. 2014;28:168-180.
36. Liland KH, Stefansson P, Indahl UG. Much faster cross-validation in PLSR-modelling by avoiding redundant calculations. J Chemom. 2020;34:e3201.
37. Indahl UG, Liland KH, Næs T. Canonical partial least squares—a unified PLS approach to classification and regression problems. J Chemom. 2009;23(9).
38. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: 32nd International Conference on Machine Learning ICML 2015. Lille, France; 2015.
39. Kristoffersen KA, Liland KH, Böcker U, Wubslet SG, Lindberg D, Horn SJ, Afseth NK. FTIR-based hierarchical modeling for prediction of average molecular weights of protein hydrolysates. Talanta. 2019;205:12.
40. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;14:1929-1958.

**SUPPORTING INFORMATION**
Additional supporting information may be found in the online version of the article at the publisher's website.

**How to cite this article:** Helin R, Indahl UG, Tomic O, Liland KH. On the possible benefits of deep learning for spectral preprocessing. Journal of Chemometrics. 2022;36(2):e3374. doi:10.1002/cem.3374