Norwegian University of Life Sciences
Faculty of Science and Technology
Department of Data Science

Philosophiae Doctor (PhD)
Thesis 2023:47

Exploring the transition from traditional data analysis to machine- and deep learning approaches

Utforsking av overgangen fra tradisjonell dataanalyse til metoder med maskin- og dyp læring

Runar Helin

Exploring the transition from traditional data analysis to machine- and deep learning approaches

Utforsking av overgangen fra tradisjonell dataanalyse til metoder med maskin- og dyp læring

Philosophiae Doctor (PhD) Thesis

Runar Helin

Norwegian University of Life Sciences
Faculty of Science and Technology
Department of Data Science

Ås, 2023

Thesis number 2023:47
ISSN 1894-6402
ISBN 978-82-575-2076-2

Acknowledgments

First, I want to thank my main supervisor Kristian Hovde Liland. I am very grateful for all the guidance, patience, feedback and discussions that helped me stay motivated throughout my thesis work. I also want to thank my co-supervisors Oliver Tomic and Ulf Geir Indahl for all the interesting ideas, fruitful discussions and helpful advice. Each supervisor also deserves a special thanks for introducing me, during my studies at NMBU, to the mathematics, programming and machine learning that made me want to pursue this PhD.

I want to thank all my colleagues in the data science, physics and material science departments. The lunches, coffee breaks, and friendly chats have all been invaluable and made every day more interesting.

Finally, I want to thank my family for all the support and especially my girlfriend, Hanne, for always being encouraging and tolerating the many late working hours, especially towards the end of this thesis work.

Summary

Data analysis methods based on machine- and deep learning approaches are continuously replacing traditional methods. Models based on deep learning (DL) are applicable to many problems and often have better prediction performance compared to traditional methods. One major difference between the traditional methods and machine learning (ML) approaches is the black box aspect often associated with ML and DL models. The use of ML and DL models offers many opportunities but also challenges. This thesis explores some of these opportunities and challenges of DL modelling with a focus on applications in spectroscopy.

DL models are based on artificial neural networks (ANNs) and are known to automatically find complex relations in the data. In Paper I, this property is exploited by designing DL models to learn spectroscopic preprocessing based on classical preprocessing techniques. It is shown that the DL-based preprocessing has some merits with regard to prediction performance, but there is considerable extra effort required when training and tuning these DL models. The flexibility of ANN architecture designs is further studied in Paper II when a DL model for multiblock data analysis is proposed which can also quantify the importance of each data block.

A drawback of the DL models is the lack of interpretability. To address this, a different modelling approach is taken in Paper III where the focus is to use DL models in such a way as to retain as much interpretability as possible. The paper presents the concept of non-linear error modelling, where the DL model is used to model the residuals of the linear model instead of the raw input data. The concept is essentially a shrinking of the black box aspect since the majority of the data modelling is done by a linear interpretable model.

The final topic explored in this thesis is a more traditional modelling approach inspired by DL techniques. Data sometimes contain intrinsic subgroups which might be more accurately modelled separately than with a global model. Paper IV presents a modelling framework based on locally weighted models and fuzzy partitioning that automatically finds relevant clusters and combines the

predictions of each local model. Compared to a DL model, the locally weighted modelling framework is more transparent. It is also shown how the framework can utilise DL techniques to be scaled to problems with huge amounts of data.

Sammendrag

Metoder basert på maskin- og dyp læring erstatter i stadig økende grad tradisjonell datamodellering. Modeller basert på dyp læring (DL) kan brukes på mange problemer og har ofte bedre prediksjonsevne sammenlignet med tradisjonelle metoder. En stor forskjell mellom tradisjonelle metoder og metoder basert på maskinlæring (ML) er den "svarte boksen" som ofte forbindes med ML- og DL-modeller. Bruken av ML- og DL-modeller åpner opp for mange muligheter, men også utfordringer. Denne avhandlingen utforsker noen av disse mulighetene og utfordringene med DL modeller, fokusert på anvendelser innen spektroskopi.

DL-modeller er basert på kunstige nevrale nettverk (KNN) og er kjent for å kunne finne komplekse relasjoner i data. I Artikkel I blir denne egenskapen utnyttet ved å designe DL-modeller som kan lære spektroskopisk preprosessering basert på klassiske preprosesseringsteknikker. Det er vist at DL-basert preprosessering kan være gunstig med tanke på prediksjonsevne, men det kreves større innsats for å trene og justere disse DL-modellene. Fleksibiliteten til design av KNN-arkitekturer er studert videre i Artikkel II hvor en DL-modell for analyse av multiblokkdata er foreslått, som også kan kvantifisere viktigheten til hver datablokk.

En ulempe med DL-modeller er manglende muligheter for tolkning. For å adressere dette, er en annen modelleringsframgangsmåte brukt i Artikkel III, hvor fokuset er på å bruke DL-modeller på en måte som bevarer mest mulig tolkbarhet. Artikkelen presenterer konseptet ikke-lineær feilmodellering, hvor en DL-modell blir bruk til å modellere residualer fra en lineær modell i stedet for rå inputdata. Konseptet kan ses på som en krymping av den svarte boksen, siden mesteparten av datamodelleringen er gjort av en lineær, tolkbar modell.

Det siste temaet som er utforsket i denne avhandlingen er nærmere en tradisjonell modelleringsvariant, men som er inspirert av DL-teknikker. Data har av og til iboende undergrupper som kan bli mer nøyaktig modellert hver for seg enn med en global modell. Artikkel IV presenterer et modelleringsrammeverk basert på lokalt vektede modeller og "fuzzy" oppdeling, som automatisk finner

relevante grupperinger ("clusters") og kombinerer prediksjonene fra hver lokale modell. Sammenlignet med en DL-modell, er det lokalt vektede modelleringsrammeverket mer transparent. Det er også vist hvordan rammeverket kan utnytte teknikker fra DL for å skalere opp til problemer med store mengder data.

List of Papers

Paper I

Helin R., Indahl U. G., Tomic O., Liland K. H. On the possible benefits of deep learning for spectral preprocessing. *Journal of Chemometrics*, 36(2):e3374, 2022.

Status: Published

Paper II

Jenul A., Schrunner S., Huynh B. N., Helin R., Futsæther C. M., Liland K. H., Tomic, O. Ranking feature-block importance in artificial multiblock neural networks. In: *Artificial Neural Networks and Machine Learning – ICANN 2022*, vol. 13532, pages 163-175, Springer Nature Switzerland, 2022.

Status: Published. Reproduced with permission from Springer Nature.

Paper III

Helin R., Indahl, U. G., Tomic, O., Liland, K. H. Non-linear shrinking of linear model errors. *Analytica Chimica Acta*, 1258:341147, 2023.

Status: Published

Paper IV

Helin, R., Indahl, U. G., Tomic, O., Liland, K. H. Fuzzy regression and classification using locally weighted ensemble models (LoWEM).

Status: Manuscript, not published.

Contents

Acknowledgments i

Summary iii

Sammendrag v

List of Papers vii

List of Figures xiii

List of Tables xv

1 Introduction 1

1.1 Motivation and background 1

1.1.1 Data analysis approaches 1

1.1.2 Spectroscopic data 3

1.1.3 Related work 3

1.2 Research aims 4

1.3 Outline 5

2 Theory 7

2.1 Spectroscopy 7
2.2 Partial least squares model 10
2.3 Preprocessing 12
2.4 Fuzzy C-means 14
2.5 Machine learning modelling 15
2.6 Deep learning 16
2.6.1 Backpropagation 18
2.6.2 Convolutional neural networks 20
2.6.3 Network hyperparameters 21

3 Summary of papers **23**

3.1 Paper I - Spectral preprocessing 23
3.2 Paper II - Ranking feature-block importance 24
3.3 Paper III - Non-linear error modelling 25
3.4 Paper IV - Locally weighted ensemble modelling (LoWEM) . . . 26

4 Discussion **29**

4.1 Overview 29
4.2 Contribution 29
4.2.1 DL models for spectroscopic data 30
4.2.2 Applying DL models in a traditional framework 31
4.2.3 Improving traditional methods using DL techniques . . . 32
4.2.4 Alternatives to DL models 32
4.2.5 Transparency VS performance 33
4.3 Future perspectives 34

Paper I 39

Paper II 61

Paper III 77

Paper IV 91

List of Figures

2.1 Example of NIR spectra from corn. 8

2.2 Example of FTIR spectra taken from dry-films obtained during enzymatic protein hydrolysis of different rest raw materials and commercial enzymes. 9

2.3 Example of Raman spectra from samples of milk. 9

2.4 Score and loading plot of a PLS model fitted to NIR spectra of corn with moist content as response value. The NIR spectra were preprocessed using a Savitzky-Golay filter width window size 9, 2nd order polynomial and 2nd derivative. 11

2.5 Illustration of how deep learning relates to machine learning and artificial intelligence. 16

2.6 Illustration of the connection of one layer neural network. The white node illustrates the bias unit. 18

2.7 Illustration of a simple multi-layered Perceptron (MLP) network. The white nodes illustrate the bias unit. 19

2.8 Illustration of the cross-correlation (flipped convolution) between a signal and a filter. 20

4.1 Illustration of how the different works in this thesis are tied together in the context of traditional and modern data analysis approaches. 30

List of Tables

2.1 Some examples of deep learning model parameters to tune . . . 22

Chapter 1

Introduction

1.1 Motivation and background

In recent years deep learning (DL) models have become the dominant tool in natural language processing and image recognition tasks. The DL models, trained on a huge amount of data, achieve remarkable results compared to more traditional methods on these tasks. In essence, the DL models are just complicated networks of interconnected nodes that learn to map data from an input domain to an output domain. When trained on enough data, this mapping can be very accurate, but the details of exactly how the DL model relates the input to the output are lost in the complicated network structure. DL models are gradually replacing traditional methods in different domains, and this transition is the subject of this thesis. Both possibilities and challenges of DL modelling are explored with a focus on applications in spectroscopy.

1.1.1 Data analysis approaches

The work in this thesis has focused on exploring the transition from traditional data analysis to machine- and deep learning approaches. This topic is quite broad and could have been taken in many different directions. Therefore, a more precise description of the chosen scope of this thesis is given here. The first point is to define what is meant by *traditional data analysis* and *modern machine- and deep learning approaches* in the context of this thesis.

Machine learning (ML) can be seen as a collection of techniques which uses an algorithm to adjust model parameters based on some input data and an

objective (i.e. cost-function/loss-function) [1]. In other words, provided with some data, a machine learning algorithm tries to learn how to perform a particular task. Machine learning covers a broad range of models including DL methods.

On the other hand, the term traditional data analysis is taken to cover methods and techniques which have a focus on the interpretation of models and data. These methods cover linear models and utilisation of domain knowledge to simplify the problems (i.e. specific preprocessing of data). Such methods are often based on assumptions and simplifications in order to utilise linear relationships between variables, which makes interpretation easier. Much research in this thesis has been done using spectroscopic data. Therefore, the discipline of chemometrics is taken as part of the traditional analysis methods. Chemometrics is the study of chemical data using mathematical, computational and statistical tools such as multivariate data analysis, design of experiments and curve fitting.

Admittedly, the border between machine learning methods and traditional analysis methods is fuzzy. For example, models such as basic linear regression and principal component analysis (PCA) are used in both disciplines. However, the use of these models may differ. A machine learning person might use the PCA model for dimensionality reduction with the aim of reducing noise and improving the prediction performance. In chemometrics, on the other hand, the principal components from the PCA would be analysed in more detail to look for patterns in the data in relation to the variation explained by the principal components.

To summarise, the main difference between traditional methods and machine learning is that machine learning is mainly focused on prediction performance. This focus has steered the development of new models in increasingly complex models to model the data more accurately, which has led to the recent success of DL models and artificial neural networks (ANNs). These models are powerful tools capable of solving highly non-linear problems. However, the complex inner structure of these models makes them black boxes with few options for interpreting the relation between input and output. More traditional approaches can be interpreted but they lack the same predictive power as the modern DL models. The gap between traditional data analysis and modern ML approaches is the subject of study in this PhD project.

1.1.2 Spectroscopic data

To explore the transition from traditional data analysis to modern machine and deep learning approaches, a field of study where both traditional and machine learning approaches have some merits was chosen. This excluded image recognition and natural language processing (NLP) tasks, where DL models clearly outperform more traditional models and are able to utilise more data than traditional models can handle. The majority of the research in this thesis was therefore limited to focus on challenges and opportunities in spectroscopy. During the time at the start of this thesis work (2019), DL models showed promising results in data modelling of spectroscopic data [2, 3, 4]. Still, the research on the topic in the literature was limited. Spectroscopy thus offered an ideal field to explore the transition from traditional methods to machine learning approaches. A typical spectroscopic dataset encountered in chemometrics has only up to a few hundred samples. Furthermore, the data often contains more features than samples, which can pose a challenge for DL models which are often trained on massive datasets. Spectroscopic data is also challenging to model because of its high dimensionality and multicollinearity. Tools from chemometrics efficiently deal with these challenges, but the question is whether machine learning approaches can be equally or even more efficient.

1.1.3 Related work

This thesis contributes to the development of new modelling techniques for spectroscopic data and provides new tools for chemometricians to use. The contribution will be an extension to other research done on the subject. A short summary of some related work is given here.

The use of ANNs in chemometrics can be traced back to the early 90s [5]. The ANN models at the time were simpler compared to modern ANN architectures and had limited applications at the time. Later, convolutional neural networks showed remarkable results for image classification tasks [6, 7, 8], and when it turned out that similar architectures were favourable on spectroscopic data, the interest of DL models in spectroscopy began to increase. The CNNs were first developed for image recognition problems where one key aspect is the combination of multiple convolutional filters to detect features in the image independent of its spatial locations. Despite this aspect not being necessary for 1D spectroscopic data, the same CNN architecture has been shown to be a good choice also in spectroscopy. Compared to traditional chemometric models such as partial least squares (PLS), which operates in a lower-dimensional subspace, the CNN models have been successfully applied with even tens of thousands of

model parameter to train, suggesting that there might be some redundancy in the CNN model weights. The use of CNNs for spectroscopy is partly motivated by the assumption that the convolutional filters are suitable to extract different properties from peaks and shapes from the spectra.

Other research on the use of DL models in spectroscopy includes the use of autoencoders. An autoencoder is a kind of ANN model which simultaneously learns an encoding of a signal to a lower-dimensional latent space and a decoding of the lower-dimensional representation so that most of the original signal is reconstructed. The autoencoder can also be used to decode the lower-dimensional signal into other signals related to the input. For example, an autoencoder can be trained to learn a computationally demanding preprocessing procedure [9] or predict future spectra in a chemical process [10]. A different approach involves the use of data augmentation techniques to artificially increase the sample size of the spectroscopic data. For example, an augmentation technique based on extended multiplicative signal correction (EMSC) has been proposed [11] and the master student Tahla Neveed also explored the effects of different data augmentation techniques on spectroscopic data [12] under my supervision.

1.2 Research aims

The overarching aim of this thesis is to explore different modelling techniques to build the bridge between traditional data modelling and modern machine learning technology. On one hand, the possibilities of using DL technology to improve traditional models in terms of prediction accuracies are studied. On the other hand, how traditional methods can be used in conjunction with DL models is explored. The focus of the thesis is on applications in spectroscopy with an emphasis on exploring different alternatives to pure black box ANN modelling. More specifically, the thesis is concentrated on the specific research goals given below.

1) Explore techniques to improve DL models for typical short-wide spectroscopic data.

2) Develop methods that use DL models in a more traditional data modelling framework.

3) Develop methods based on DL techniques to improve the predictive performance of traditional models while retaining interpretability.

4) Develop alternatives to DL models based on traditional methods.

1.3 Outline

In Chapter 2, a short introduction to spectroscopic data analysis is given followed by a presentation of chemometric- and machine learning approaches. Chapter 3 presents the results as a summary of the papers making up this thesis followed by a discussion in Chapter 4. The full articles are included at the end.

Chapter 2

Theory

In this section, the relevant methods and tools studied in the thesis are presented. First, a general overview of spectroscopy is given followed by a description of the partial least squares (PLS) model and common preprocessing techniques for this kind of data. Then, the Fuzzy C-means (FCM) algorithm is presented followed by an introduction to machine learning and deep learning (DL) models with a description of some common types of artificial neural network (ANN) models.

2.1 Spectroscopy

Spectroscopy is the study of interactions between matter and electromagnetic radiation. Properties such as absorbance, transmission, reflection and emission are studied. When analysed, these properties can give information about the structure of the sample at the atomic level. Different types of interactions and measuring techniques result in different kinds of spectroscopic data to study. Three types of spectra used in this thesis, all in the category of vibrational spectroscopy, are near-infrared (NIR), Fourier-transform infrared (FTIR) and Raman spectra.

These kinds of spectroscopic data are obtained by the study of molecular vibrations. Each molecule has characteristic interactions with electromagnetic radiation (light) of different frequencies/wavelengths. By sending light with different frequencies to a sample and measuring these interactions, the corresponding spectrum is obtained. By studying the peaks in the resulting spectrum, it is, for example, possible to say something about the chemical compounds in the sample. A spectrum typically contains information about hundreds or even

thousands of wavelengths, making the data extremely high-dimensional.

The NIR spectra show the absorbance of light as a function of the wavelength of light. The wavelengths lie in the range between 700 nm to approximately 2500 nm and cover the infrared part of the light spectrum closest to the visible light region, hence the name NIR. The spectra are obtained by sending light with different wavelengths to a sample and calculating the absorbance by comparing the incidence light with the light reflected or transmitted from the sample. Compared to the other spectra, the NIR spectra are characterised by broad and overlapping peaks and are less specific as illustrated in Figure 2.1.

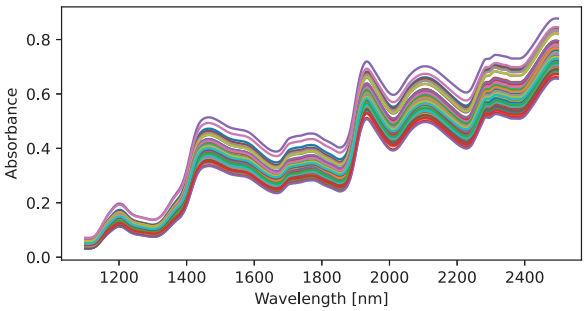


Figure 2.1: Example of NIR spectra from corn.

The Fourier transform infrared (FTIR) data is an alternative to NIR data. One difference is that the FTIR spectra cover a broader range of wavelengths and typically have a higher resolution. An example is shown in Figure 2.2 where the absorbance is plotted as a function of the frequency instead of wavelength, as is typically done for this type of spectra. They typically measure wavelengths in the NIR and mid-infrared region of the light spectrum up to 25 µm, but larger wavelengths are also possible. When obtaining the FTIR spectra, light consisting of the whole desired wavelength range is sent to the target. With a beam splitter, two beams that have travelled different optical path lengths through the sample are then combined. Based on interference patterns between the two beams and Fourier transformation, the FTIR spectra can be calculated.

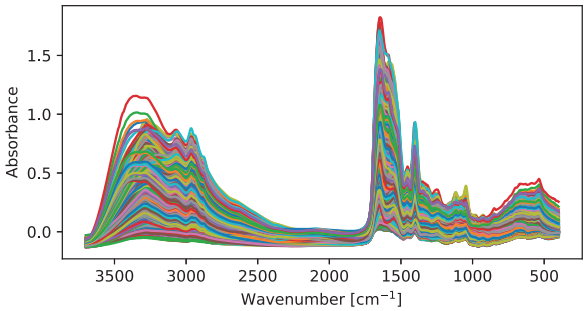


Figure 2.2: Example of FTIR spectra taken from dry-films obtained during enzymatic protein hydrolysis of different rest raw materials and commercial enzymes.

Raman spectra follow a different principle than NIR and FTIR. It is not based on the absorbance of light but on inelastic scattering effects. The spectra are obtained by sending monochromatic light with different frequencies on a sample. Most of the scattered light from the sample will have the same frequency as the incident light, but a small amount has a change in frequency. At each measured wavelength, the intensity of this Raman scattering forms a Raman spectrum. This kind of data is often regarded as complementary to FITR spectra since it measures molecular vibrations that do not absorb infrared light and are thus not captured in the FTIR spectra. Usually, the Raman spectra contain signals caused by fluorescence which need to be separated out later. The Raman spectra are often characterised by having peaks of high resolution in contrast to NIR spectra for example as illustrated in Figure 2.3. Similar to FTIR spectra, the Raman spectra are depicted using frequencies instead of wavelengths.

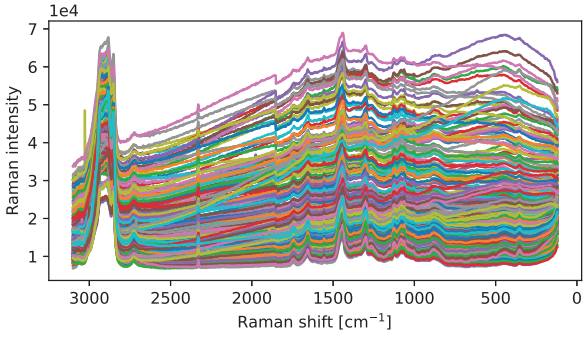


Figure 2.3: Example of Raman spectra from samples of milk.

Spectroscopic data analysis

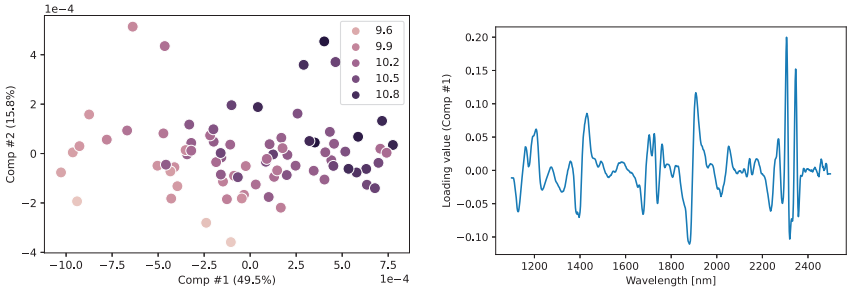
While the spectra can be analysed directly to determine the chemical constituents in a sample they can also be used to predict a response value or to classify samples into different groups. A successful model requires high-quality measurements and good reference values. The process of collecting these reference values is often time-consuming and needs to be obtained through chemical or physical processes. They often rely on manual work and can destroy the sample in the process. The investment in obtaining high-quality reference values pays off when having obtained an accurate model, which repeatedly can make predictions quicker and non-destructive from the measured spectra.

Beer-Lamberts law states that absorption is proportional to the concentrations of chemical analytes. This law is a common assumption in chemometrics and is a justification for the extensive use of linear models such as partial least squares (PLS) models. Although Beer-Lamberts law often is valid, spectroscopic data can sometimes be distorted by so-called interferents, which make the data more non-linear [13]. Other sources of nonlinearities include noise and complex or heterogeneous sample structures. To successfully model the spectroscopic data with such non-linearities a single linear model is not sufficient. Variations of the PLS model such as kernel PLS are one option to model non-linearities [14]. The fact that some spectroscopic data contain non-linearities is the reason DL has successfully been used to model such data.

2.2 Partial least squares model

The partial least squares (PLS) model is a commonly used model in chemometrics [15]. The model is well suited for data suffering from high multicollinearity and solves this issue by a dimensional reduction. In contrast to other dimensionality reduction methods such as principal component analysis (PCA), the features in the new lower dimensional latent space, called components, are found with the idea of best predicting the response $\boldsymbol{y}$ from the data $\boldsymbol{X}$. More specifically, the components are found as explaining most of the covariance between $\boldsymbol{X}$ and $\boldsymbol{y}$.

The goal of the PLS algorithm is to decompose $\boldsymbol{X}$ as $\boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E}$, where the columns in the matrices $\boldsymbol{T}$ and $\boldsymbol{P}$ are called the scores and loadings respectively and $\boldsymbol{E}$ is the residual matrix. There are also corresponding loading weights $\boldsymbol{W}$ and y-loading $\boldsymbol{q}$. There exist different PLS algorithms with differences in calculation speed and numerical stability.



(a) Scoreplot of the first two PLS components

(b) Loading plot of the first PLS component

Figure 2.4: Score and loading plot of a PLS model fitted to NIR spectra of corn with moist content as response value. The NIR spectra were preprocessed using a Savitzky-Golay filter width window size 9, 2nd order polynomial and 2nd derivative.

Regression coefficients can then be calculated based on the m components corresponding to the highest covariance to obtain a partial least squares regression (PLSR) model with the formula

$$\boldsymbol{\beta}_{PLS} = \boldsymbol{W}\left(\boldsymbol{P}^T\boldsymbol{W}\right)^{-1}\boldsymbol{q} \tag{2.1}$$

The scores and loadings in the PLS model are useful for interpretation and can give you insights into the dataset. Figure 2.4 shows an example of a score- and loadings plot from a PLS model trained to predict moist content from NIR spectra of corn. In the score plot to the left, The x and y axis are the two first PLS components respectively with the percentage of explained variance for each given. The moist content of each sample is indicated by different colours. It can be seen that the samples with less moisture tend to be located to the left while the samples with higher moist content are more to the right. This suggests that the variance in moist content in the sample is clearly captured by the first PLS component. From the loading plot, the peaks indicate which original features the PLS model focuses on. It can be used to determine what chemical properties the component focuses on. For instance, the peaks at around 2300 nm correspond to wavelengths with high absorption of oil while peaks at 1900 and 1450 are bands with large water absorption [16].

The score and loadings plot are powerful ways to gain information about the dataset. It can be used to infer the types of variance the PLS components explain as well as detect groups of data that share similarities.

The PLS model can also be used for classification in binary or multiclass problems. It is done by using a dummy-matrix representation (one-hot encoding)

of the class-membership vector and using the PLS2 algorithm suitable for multiple responses to fit the data. Alternatively, the PLS-DA [17] algorithm can be used. In both cases, the PLS model is combined with a discriminant analysis such as linear discriminant analysis (LDA) to obtain the class predictions.

2.3 Preprocessing

Preprocessing of data is usually a required step in data analysis in order to get meaningful data for the model to train on [18]. Common preprocessing steps include transformations of features, feature selection, handling missing values, mean centring and normalisation of each feature in the data and outlier detection. Different domains also have specific preprocessing steps to handle known effects present in the data. In the field of spectroscopy, preprocessing steps aim at removing variation in the data caused by unwanted physical effects. For example, some preprocessing techniques correct for multiplicative effects caused by differences in the optical pathway when measuring a sample. The differences make the light travel different lengths resulting in uneven light absorption in the sample. This is among other things caused by differences in material thickness, heterogeneity in the sample or variation in the light source. The standard preprocessing techniques in spectroscopy are designed to correct for similar known effects. In the following, a few such preprocessing techniques are presented.

Multiplicative scatter correction: Multiplicative scatter correction model (MSC) is an example of a specific preprocessing model used in vibrational spectroscopy. The method was first developed for the analysis of NIR spectra [19]. Based on the Beer-Lambert law, the MSC model decomposes each spectrum $\boldsymbol{x}$ as a constant baseline a, a typical spectrum $\boldsymbol{x}_{ref}$ called a reference spectrum scaled by a constant b and an error term $\boldsymbol{e}$. More specifically, each spectrum $\boldsymbol{x}_i$ is decomposed as

$$\boldsymbol{x}_i = a + \boldsymbol{x}_{ref} \cdot b + \boldsymbol{e}, \tag{2.2}$$

where, $\boldsymbol{e}$ is a residual term. The scalars a and b are unique for each sample but found using a common reference spectrum. The parameters a and b are estimated using a least squares approximation. The idea is to make all samples as close to the reference spectrum as possible by removing the additive and multiplicative effects. The interesting chemical variation in the spectrum is then left in the error term $\boldsymbol{e}$. After obtaining estimates of a and b, a corrected spectrum $\boldsymbol{x}_{corr}$ is given by

$$\boldsymbol{x}_{corr} = \frac{\boldsymbol{x}_i - a}{b} \tag{2.3}$$

Extended multiplicative signal correction: By taking the MSC model as a basis, the extended multiplicative signal correction (EMSC) model was developed as a generalisation of the MSC model where the same framework is used to correct for more general signals in addition to scattering effects [20, 21]. The model is based on the same derivation but adds more terms corresponding to basis vectors of polynomials $\boldsymbol{\nu}^j$ with increasing polynomial degree. The corrected spectra for en EMSC model with $j = 1,\ldots, n$ polynomial correction is defined by

$$\boldsymbol{x}_{corr} = \frac{\boldsymbol{x}_i - a - \sum_{j=1}^{n} d_j \boldsymbol{\nu}^j}{b} \tag{2.4}$$

Both the MSC and EMSC corrected spectra can be written in an alternative form

$$\boldsymbol{x}_{corr} = \boldsymbol{x}_{ref} + \frac{1}{b}\boldsymbol{e} \tag{2.5}$$

In this form, it is more apparent that the variation in the spectra is caused by chemical factors captured in the residual term $\boldsymbol{e}$, and that the corrected spectra correspond to variation around a common reference spectrum. Usually, the mean spectrum across all spectra is taken as the reference spectrum.

The EMSC model using second-order polynomial correction is often called basic EMSC. Additionally, numerous variations of the EMSC method have been developed such as the ability to include corrections of known physical or chemical variations with the use of constituent spectra [20], add replicate correction [22] or correct for Mie-scattering [23].

Standard normal variate: A closely related preprocessing technique to MSC is the standard normal variate (SNV)[24]. The SNV correction is identical to Equation 2.3. However, the scalars a and b are instead taken to be the mean value and standard deviation of spectrum $\boldsymbol{x}_i$. In contrast to MSC, the SNV corrects each spectrum individually and is not based on a common reference spectrum.

Savitzky-Golay: Another popular preprocessing technique is the Savitzky-Golay (SG) filtering [25]. This method aims at smoothing a signal without

distorting the information it contains. It achieves this by fitting a polynomial curve to a local neighbourhood around every point and using the local polynomial approximation as the smoothed signal. This procedure can be seen as a convolution of the signal with a kernel representing the polynomial approximation. In addition to smoothing a signal, SG filtering can be used to approximate derivatives of the signals, which has been proven useful for noisy types of signals such as in FTIR spectroscopy [26].

In the context of PLS models, preprocessing has the benefit that one typically needs fewer components to successfully model the signal. In other words, less noise is included in the model if preprocessing is performed. The same argument can be made for neural networks as well, and in that sense, preprocessing should not be disregarded even though it might not be strictly necessary for good prediction performance. It can, on the other hand, improve robustness.

2.4 Fuzzy C-means

Fuzzy C-means (FCM) is an iterative clustering algorithm similar to K-means clustering [27]. The difference is that instead of assigning each sample to a cluster, each sample is given a set of membership values between 0 and 1, one for each cluster in the algorithm. For each sample, these membership values sum to one across all clusters and represent how similar the sample is to a prototype of each cluster or the distance to the cluster centres. The clustering is fuzzy since a sample belongs, to some degree, to more than one cluster. The amount of fuzziness is controlled by a parameter $q > 1$ which affects the clustering. A value close to 1 translates to very crisp clusters, where samples have large membership values for the closest clusters and small values for the others, while large values of q mean that the membership values are almost equal for every cluster.

The FCM algorithm starts with initialising $k = 1, \ldots, m$ cluster centres θ_k either randomly or by using a more clever initialisation such as kmeans++. The membership values μ_{ik} for each sample i and cluster k are calculated according to the equation

$$\mu_{ik} = \left[\sum_{j=1}^{m} \left(\frac{||\boldsymbol{x}_i - \theta_k||^2}{||\boldsymbol{x}_i - \theta_j||^2} \right)^{\frac{1}{q-1}} \right]^{-1}, \tag{2.6}$$

Based on these membership values, new cluster centres θ_k are calculated according to

$$\theta_k = \frac{\sum_{i=1}^{n} \mu_{ik}^q \boldsymbol{x}_i}{\sum_{i=1}^{n} \mu_{ik}^q}. \tag{2.7}$$

This process of calculating the cluster membership values and updating cluster centres continues until the cluster centres stop changing or a maximum number of iterations are reached.

2.5 Machine learning modelling

The topic of this thesis has been the transition from traditional data analysis to machine learning approaches. Therefore, a more in-depth description of machine learning is given first.

As mentioned in the introduction, machine learning is a collection of techniques/models that automatically adjust model parameters based on input data and an objective function. Examples of tasks suitable for machine learning include classification, regression, identifying groupings/clusters among measured samples or even playing chess. A distinction is made between different types of learning tasks. Many tasks are of the type *supervised learning* which covers all problems of learning a mapping from some input to an output based on a labelled set of samples. The goal of such tasks is to learn this mapping from a limited (random) selection of samples to be accurate for predictions of new samples. Regression and classification tasks fall both under this category.

A second type of learning task is *unsupervised learning* which deals with the problems of discovering patterns and structure in input data without being told what to look for. Models and algorithms that do not rely on response values are considered unsupervised. Principal component analysis (PCA), and different clustering algorithms, such as fuzzy C-means, are examples of this type of learning task. Autoencoders are examples of unsupervised DL models.

A third type of learning task is *reinforcement learning* which is a type of learning similar to trial and error. Given a defined environment, a learning algorithm (also called an agent) is given the challenge of learning the optimal choices in different situations within this environment. This type of learning is the driving force of the recent AlphaZero-machine which learned to play chess, shogi and go on a super-human level [28]. In this thesis, most of the work is done relating to supervised learning and some unsupervised learning.

Machine learning approaches are driven by making accurate models. When faced with a modelling challenge, the choice of model and corresponding hyper-

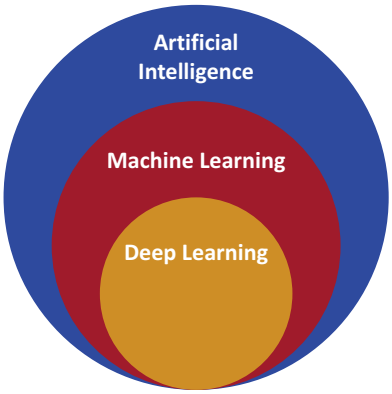


Figure 2.5: Illustration of how deep learning relates to machine learning and artificial intelligence.

parameters need to be chosen to fit the particular problem. The process is summarised to first find an appropriate model. The second step includes an evaluation of how well the model performs. Depending on the size of the dataset, a typical approach is to divide the dataset into a training and test partition, where the test partition is used to estimate the model performance. This partition will only be used when the best model is already chosen. The model needs to be chosen based on the training set. The typical approach is to use a part of the training data as a validation set to evaluate different model and hyperparameter choices. Alternatively, a cross-validation procedure can be performed. The process of finding the optimal model in this framework is usually automated and the main concern is to minimise the prediction error.

2.6 Deep learning

Deep learning is a sub-field of machine learning and consists of machine learning models which are constructed to learn different layers of representations from the input data [29]. In a broader sense, machine learning is again a sub-field of artificial intelligence as illustrated in Figure 2.5.

Deep learning models are built from interconnected nodes which make up what is called an artificial neural network (ANN), also referred to as a deep neural network (DNN). The layers of representations are called *hidden layers* and the number of hidden layers in a network is referred to as the depth of the network.

Mathematically speaking, a deep learning model is the mapping $f_\Theta : \mathbb{R}^p \to \mathbb{R}^c$ from some input data $\boldsymbol{x} \in \mathbb{R}^p$ to an output $\boldsymbol{y} \in \mathbb{R}^c$, where Θ is the model parameters to be learned. The mapping is a composite function of intermediate mappings where each intermediate mapping corresponds to a layer in the network. Each layer l is described by a non-linear function $f^{(l)}(\boldsymbol{z}^{(l-1)})$, where $\boldsymbol{z}^{(l-1)}$ is the output of layer $l-1$. The whole deep learning model with d number of layers is thus the composition of such non-linear functions on the form:

$$f_\Theta(\boldsymbol{x}) = f^{(d)}(f^{(d-1)}(\cdots(f^{(1)}(\boldsymbol{x}))\cdots)) \tag{2.8}$$

The challenge of deep learning models is to learn these layer representations given a set of data samples $(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, \ldots, n$.

Fully connected network

The most basic neural network is the fully connected feed-forward network also known as a multilayer Perceptron (MLP) or a dense network. These MLP models are all built from individual nodes called *neurons*. Each neuron receives a vector as the input signal and returns a single scalar as the output. The output z of a neuron is based on an affine function of its input $\boldsymbol{x}$ on the form

$$z = \sigma\left(\boldsymbol{x}^T\boldsymbol{w} + b\right), \tag{2.9}$$

where $\boldsymbol{w}$ and b are the neuron parameters called the weights and bias respectively. The function $\sigma(\cdot)$ is a non-linear scalar function called the activation function, and is the key factor that makes the network able to model non-linearities. There are a lot of different choices for activation functions, but one of the most common is the rectified linear unit (ReLU) defined by

$$\sigma(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

A slight variation of the ReLU activation function is the exponential linear unit (ELU) defined by

$$\sigma(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{otherwise} \end{cases} \tag{2.11}$$

Compared to the ReLU activation function, ELU is supposed to achieve faster convergence and better prediction accuracy [30].

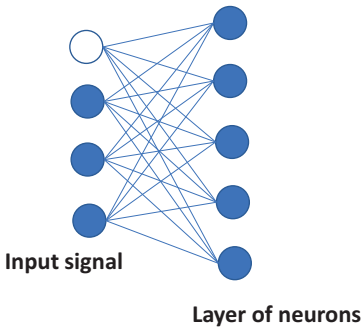


Figure 2.6: Illustration of the connection of one layer neural network. The white node illustrates the bias unit.

When several neurons are connected to the same input signal, a layer in the network is formed as shown in Figure 2.6. In the figure, an input signal of length 3 is connected to a layer containing 5 neurons. This layer is called fully connected since all the neurons are connected to each value in the input signal.

The set of outputs of all the neurons in the hidden layer is the layer representation of the input and can be connected to another hidden layer to form a new layer representation of the input. Figure 2.7 shows a network with two hidden layers plus an output layer containing only one node.

Each neuron in the network contain its own set of weights and bias and the combined weights and biases of all neurons in the network forms the full set of network model parameters Θ. The network in Figure 2.7 is just an illustration of a basic ANN model with two hidden layers. The largest types of ANN models used today use hundreds of hidden layers resulting in billions of parameters.

2.6.1 Backpropagation

Backpropagation [31] is the main technique used to fit a DL model to data. Consider a neural network with weights $\boldsymbol{\Theta}$ and denote the prediction of a dataset $\boldsymbol{X}$ as $\hat{\boldsymbol{y}} = f(\boldsymbol{X}; \boldsymbol{\Theta})$. The evaluation of the prediction performance of the network is done by the loss function $L(\boldsymbol{y}, \hat{\boldsymbol{y}})$. The backpropagation algorithm is based on the *gradient descent*, which is an optimisation algorithm that iteratively updates the weights $\boldsymbol{\Theta}$ to minimise the loss function L. The gradient of the loss function with respect to the model weights gives the direction

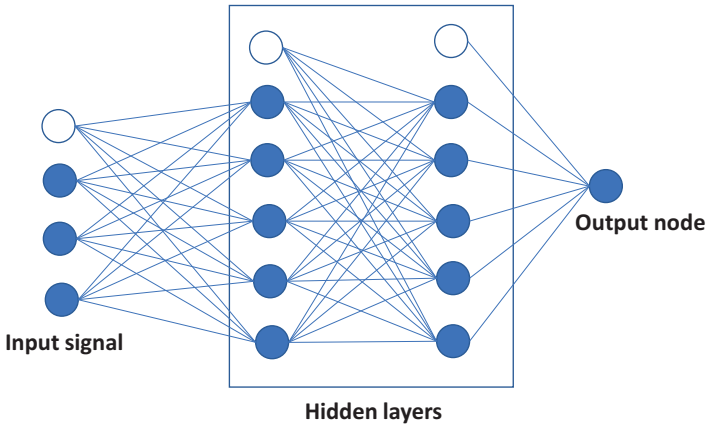


Figure 2.7: Illustration of a simple multi-layered Perceptron (MLP) network. The white nodes illustrate the bias unit.

to adjust the weights to minimise the loss function. By iteratively adjusting the weights in small steps in the gradient direction with the step size controlled by a learning rate parameter η, a minimum value of the loss function will be found. The optimisation is highly non-convex so it is impossible to know if it is the global minimum (i.e. the best solution one can find). Let the gradient with respect to the weights from a hidden layer h at iteration number t be denoted $\nabla L_{\boldsymbol{W}_h^{(t)}}$. Using the gradient descent, the weights at the next iteration $t+1$ are given by

$$\boldsymbol{W}_h^{(t+1)} = \boldsymbol{W}_h^{(t)} - \eta \nabla L_{\boldsymbol{W}_h^{(t)}} \tag{2.12}$$

Backpropagation is an efficient algorithm to update the ANN model by first calculating the gradient of the final layer and subsequently calculating the other gradients moving backwards through the network. This is sometimes called error propagation. The reason this is efficient is the structure of the ANN model as a composition of functions. Therefore the gradient at one layer can be calculated using the chain rule starting from the final layer of the network. By starting with the final layer, the calculations for the errors can be reused in the calculation from the earlier layers.

The neural network training is usually performed by stochastic gradient descent where smaller random subsets of the data, called mini-batches, are used to calculate the gradients at each iteration instead of using the whole dataset. This allows the ANN models to be trained on extremely large datasets because the whole dataset does not need to be stored in computer memory at the same

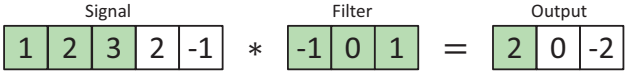


Figure 2.8: Illustration of the cross-correlation (flipped convolution) between a signal and a filter.

time to train the model, which is usually necessary for traditional models. By utilising stochastic batch-wise training with small weight updates, the network weights usually converge towards good values for minimising the loss function.

2.6.2 Convolutional neural networks

Convolutional neural networks are deep learning models in the same form as equation 2.8, but where at least one of the intermediate mappings $f^{(l)}$ is replaced with the convolution operator. Mathematically, a convolution of two signals $\boldsymbol{u}$ and $\boldsymbol{v}$ with lengths N and M respectively is defined by

$$[\boldsymbol{u} \circledast \boldsymbol{v}](i) = \sum_{j=1}^{N} u_j v_{i-j}, \quad \text{for } i = 1, \ldots, M \tag{2.13}$$

In the context of ANNs, the convolution is performed between the input signal $\boldsymbol{x}$ and a filter or kernel $\boldsymbol{w}$. In practice, cross-correlation is usually the value that is calculated in ANNs. The only difference between cross-correlation and convolution is that the filter is flipped. The term convolution has become standard to use in the literature. Therefore, this term will be used here even though the operation described strictly speaking is the cross-correlation.

Figure 2.8 shows the calculations between a signal and a filter. The convolutional filter is applied at every location of the input signal in a sliding fashion, calculating a sum of the elementwise multiplication of the signal and filter values at each location as the output. The operation is a kind of local correlation measure where large output values indicate a high correlation between the filter and the signal at that point and negative values indicates a negative correlation. On spectroscopic data, the convolutions are therefore useful for detecting peaks and shapes in the spectra.

In a neural network layer, the convolutional filter is the weights associated with the convolutional layer. A convolution can be represented as a matrix

multiplication with a sparse matrix replacing the convolutional operator. The weights of a convolutional layer can therefore still be represented as a matrix $\boldsymbol{W}$, making the equations for a CNN equivalent with an MLP. The convolutions are often combined with pooling operations such as max-pooling. The pooling operators are used to reduce the dimension of the signal for the next layer in the network (usually by half). Max-polling returns the local maximum value at each location of the signal. For example, by returning the maximum value of every two neighbouring values, the max-pooling operator is reducing the signal by half. In 2-dimensional (2D) CNNs for images, the use of several CNN-pooling stacks makes the network able to detect objects and specific features in the image independent of the location in the image. The CNNs for images are usually structured as having the convolutional layers first in the network working as a feature extractor. These features are later connected using fully connected layers at the end. The learned feature extractors can then be used to extract useful features on other images not being part of the network training, which is the basis of transfer learning.

In convolutional layers, the weights of the convolutional filters are updated using backpropagation just as for MLP models described above. Another benefit of CNNs is the aspect of weight sharing. Each convolutional filter is used across the whole signal which drastically reduces the number of parameters compared to a fully connected layer which has a unique weight associated with each location in the signal.

2.6.3 Network hyperparameters

When training an ANN model, the choice of architecture is crucial. The network suffers from the same balancing problem between over- and underfitting as other machine learning models. However, freedom in the hyperparameter choices is much greater. Again, if the network is too simple, i.e. too few layers, it may fail to model the relevant relation between the data and the target, but if the network is too complex it will quickly overfit the data. Tuning a network is thus an important aspect of DL modelling.

Over the years, there has been developed a number of standard network base architectures for image and text analysis models such as Unet, Resnet and Transformer networks [32]. Of course, these architectures should also be tuned before training them on new data. However, these have been proven to perform well on a certain type of data which reduces the number of hyperparameters to adjust. Deep learning models for spectroscopy are not as well studied in the literature, and no standard model for such data is established.

Table 2.1: Some examples of deep learning model parameters to tune

Category	Parameter choice
Network architecture	Activation functions
	Number of hidden layers
	Number of convolutional layers
	Size of convolutional filters
	Dropout rate
	Batch Normalisation
Network training	Optimisation algorithm
	Learning rate
	Batch size
	Number of epochs
	Weight initialisation

Instead of describing every possible choice and their implications on the network, an overview of typical parameters one might want to adjust is given in Table 2.1. Going systematically through all of them is too time-consuming given the vast number of different possible combinations, so a more shallow search based on prior experience or a Bayesian optimisation approach can be taken instead [33].

Chapter 3

Summary of papers

3.1 Paper I - Spectral preprocessing

On the possible benefits of deep learning for spectral preprocessing

Preprocessing is a mandatory step in spectroscopic data analysis to remove unwanted variations in the signals. More specifically, the preprocessing aims at removing physical effects captured by the sensors during the measurements which distort the chemical signal of interest. Choosing the correct preprocessing procedure is necessary for the successful modelling of the signal. Traditionally, the preprocessing and data modelling is performed in subsequent steps. The optimal preprocessing procedure is found by trial and error from a list of suitable preprocessing methods. This process can be time-consuming and it is not guaranteed that the best preprocessing technique is found. An alternative is to integrate preprocessing into the data modelling procedure. Artificial neural networks (ANNs) are powerful machine learning models capable of automatically finding complex relations in the data. The flexibility of the ANN model opens up the possibility to design an ANN with an integrated adaptive preprocessing procedure, which can alleviate the burden of manually finding the optimal preprocessing for each problem.

The main ambition of this paper was to explore different designs of ANN layers for spectroscopic preprocessing and how these layers affected prediction performance. One aspect was to see if the adaptive preprocessing gave an increased prediction performance in ANN models and partial least squares (PLS) models. Another aspect was to see if the preprocessing module in the

ANN model made it easier for the model to learn from fewer data. This study was inspired by a paper by Dong et al. [34], who proposed a convolutional neural network where two of the layers were restricted such that the output of these layers performed baseline correction and smoothing respectively. Our study expands on this concept by implementing the baseline correction and denoising layers as separate modules to be used with any ANN architecture. Additionally, a novel preprocessing ANN layer based on the extended multiplicative signal correction (EMSC) method was developed called neural network EMSC (NN-EMSC). In comparison to the traditional EMSC method, NN-EMSC uses an adaptive reference spectrum, where the values of the reference spectrum are adjusted according to the ANN prediction error minimisation.

In the paper, it is shown that ANN-based preprocessing can be used to obtain a slight increase in predictive performance for both PLS models and ANN models compared to classical preprocessing techniques. However, the increased computational cost and effort required in tuning and training the neural networks make deep learning-based preprocessing less attractive as an alternative to classical preprocessing. Furthermore, evidence was found that suggests, despite the effort of forcing the network to perform specific preprocessing, that the role of the preprocessing modules in the ANN in the context of the whole network dynamics might be more complex than just the desired preprocessing. This does not change the possible usefulness of the proposed ANN preprocessing layers. However, interpretations of these models are not straightforward.

3.2 Paper II - Ranking feature-block importance

Ranking feature-block importance in artificial multiblock neural networks

The study of feature importance is important in trying to understand artificial neural networks (ANNs). Many studies exist that focus on ranking the importance of individual features. However, in Paper II, the focus has been to extend the concept to ranking the importance of groups of features, referred to as blocks. The features within each block are similar in the sense that they are derived from the same source or are of the same type. Ranking of the different feature blocks gives information about redundant or non-informative blocks which then can be excluded from further modelling. Furthermore, the ranking scores of the blocks can give insights into the behaviour of ANN models.

The flexibility of ANN designs allows each feature block to enter the network in

separate branches. The proposed model is called an artificial multiblock neural network. Each branch is itself a smaller network which deeper in the multiblock network are connected. The structure of the multiblock network makes it possible to use concepts from information theory to quantify the importance of each feature-block network. Three different strategies are presented in the paper and compared using both simulated data and real cases. One approach is based on established methods for ranking individual feature contributions and the two others are based on the concept of including or excluding individual blocks in the network to study the difference in information passed through the network. It was shown that each strategy successfully identified the important blocks and that each strategy has its own merit with respect to different scenarios.

3.3 Paper III - Non-linear error modelling

Non-linear shrinking of linear model errors

ANNs are powerful models capable of modelling complex non-linear relations in the data. The drawback is that these models are black boxes which makes interpretation hard or even impossible. In many situations, model interpretation is important to gain a better understanding of the problem and to build trust towards the results. When using ANN models, interpretability is sacrificed for higher prediction accuracy. To gain the performance of an ANN while simultaneously having options for interpretations, one option is to develop techniques and tools to uncover and explain the dynamics of the neural network. An alternative is to instead use a linear interpretable model as the main model and instead use an ANN to model the residuals of that linear model. This procedure allows the same interpretation of the linear model while the ANN model is used to boost the predictive performance. The approach can be seen as a decomposition of the signal into a linear part and a non-linear part, where the non-linear part is modelled by the ANN from the linear model residuals instead of the original signal. Paper III presents this error modelling framework for regression problems and shows how the framework can be extended to classification problems. Classification can be done using a dummy matrix representation (one-hot encoding) of the class-membership vector and fit an ANN to predict the residual dummy matrix. The classification is done by discriminant analysis of the sum of linear and ANN predictions. Alternatively, the ANN can be forced to learn the residuals indirectly by training the ANN as a classification model but combining the linear model predictions and the ANN output before the classification layer. It turned out that the ANN model could undermine the intention of learning the residuals by making its own

output values extremely large in comparison with the linear prediction, thereby turning in effect the residual modelling into a pure ANN model. By adding a regularisation term on the ANN output values the model behaved as intended. Similar to the results in Paper I, it is difficult to force an ANN model to behave in a specific way.

The paper shows that for sufficiently complex problems the residual modelling scheme could achieve similar prediction performance as a pure ANN model. In some cases the ANN model failed to improve the prediction performance of the linear model, meaning that the data might not contain any relevant nonlinearities or the residuals were too noisy for the network to learn from. Despite not being able to improve the prediction performance in all the cases, it did not make the prediction worse. The paper demonstrates possible interpretation options for a linear PLS model and also how the change in the residuals after the ANN modelling can be used to gain further insights. Overall, the error modelling scheme is a simple yet powerful modelling alternative. Compared to a pure ANN model, the size of the black box model is reduced since the majority of the modelling is done by a linear interpretable model.

3.4 Paper IV - Locally weighted ensemble modelling (LoWEM)

Fuzzy regression and classification using locally weighted ensemble models (LOWEM)

In many modelling situations, the data contain intrinsic subgroups caused by differences between samples, measurement techniques or other factors. To successfully model such data, it can be advantageous to model each subgroup separately. When the information about the subgroups is known, one can first train a classifier to predict the subgroups and then train local models for each subgroup based on the classification [26]. When there is no information about the underlying subgroups available, another approach must be taken. Paper IV proposes an approach based on the concept of locally weighted ensemble models (LoWEM). The proposed method was partly inspired by theories about classes of ANNs being composed of max affine spline operators (MASOs) [35]. In fact, this interpretation makes intuitive sense since most standard ANN architectures consist of just a series of matrix multiplications and simple nonlinear activations such as the rectified linear unit (ReLU) function. The MASO theory argues that the network is actually discovering the relevant subgroups internally during the model training and that new predictions are based on

similarities to these learned subgroups. LoWEM uses fuzzy clustering but includes information about the prediction error when updating cluster centres. The idea is to find local clusters where similar samples are close in feature space but also share common relations to the response, thus resulting in an accurate model. Each local model is trained using the cluster membership values as sample weights, and the final model prediction combines all local models. The concept is also expanded to tackle classification. As the choice of the type of local models, both ordinary least squares (OLS) and partial least squares (PLS) models are used in the paper. Also discussed is the use of linear Perceptron models with batch-wise training to scale the framework up to problems with huge amounts of data.

The LoWEM framework is demonstrated on real and simulated data with a focus on high-dimensional spectroscopic data. LoWEM successfully outperforms a global model on high-dimensional problems and demonstrates the possibility of accurately classifying samples that are not linearly separable by using more clusters than the number of classes to predict. It also showed that the samples near each local cluster centre in LoWEM tend to share some biological similarities.

Chapter 4

Discussion

4.1 Overview

An overview of the thesis is given in Figure 4.1. In the figure, the linear and non-linear data modelling problems are separated. One of the main motivations for using deep learning (DL) models is their ability to model complex non-linear relations. They can in principle solve linear problems as well, but that is usually done by simpler models. The figure illustrates what the main model and kind of problem in each of the four papers focus on. For instance, it is clear that the pls model has been a staple in Papers I, III and IV while ANNs have been used in Papers I, II and III. Paper I is placed in the linear problems partition despite using ANNs because the paper focuses on preprocessing methods that are linear in nature. It is also seen that Papers III and IV both lie at the border between linear and non-linear problems. In Paper III, both linear PLS and non-linear ANN is used to model a linear and non-linear partition of the data respectively, while Paper IV combines local linear models to be able to model complex non-linearities. Finally, Paper II uses a pure ANN model capable of solving non-linear problems. This paper is connected to traditional analysis methods through the use of multiblock analysis.

4.2 Contribution

The main purpose of this thesis has been to explore the transition from traditional data modelling to machine- and deep learning approaches. To this end, new methods based on combinations of techniques from both modelling

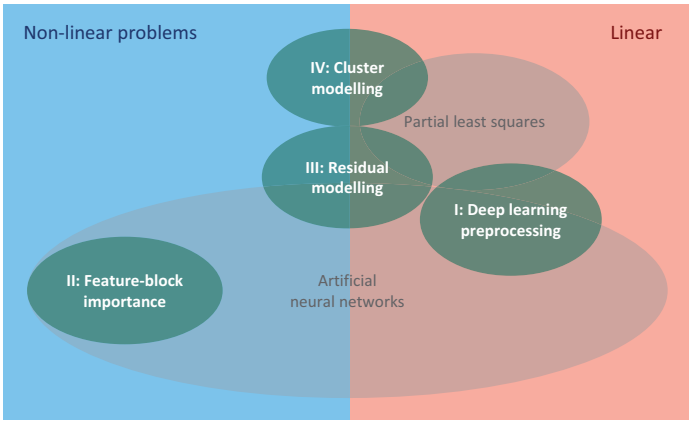


Figure 4.1: Illustration of how the different works in this thesis are tied together in the context of traditional and modern data analysis approaches.

paradigms have been developed and tested on real and simulated datasets.

4.2.1 DL models for spectroscopic data

Most of the data analysis in this thesis has been done on spectroscopic data. Therefore, a large part of the work went into exploring designs of DL models applicable for short-wide datasets typically encountered in spectroscopy. It had already been discovered that convolutional neural networks work well on these data. However, there are still many choices to make regarding the network architecture. Paper I explores to an extent an alternative multi-layer perception (MLP) architecture but they seemed to be more unstable than the ANNs based on convolutions. The CNNs used in Papers I and III for spectroscopic data were relatively shallow with just one convolutional layer followed by one to three fully connected layers. Deeper architectures were explored as well but they showed more variability in prediction performance and were more prone to overfitting. A possible reason for this behaviour could be that the more complex deeper network more easily captures the variation of the training data but fails to properly generalise. The standard approach to fix such problems would be to add restrictions to the network or train the network on more data. In fact, Paper I tries to make the problem easier for the DL models by restricting convolutional layers to perform preprocessing. The benefit of using a more shallow neural network is that the relation between the input features and the response is not as complicated which gives more hope in trying to understand this relation.

Other useful techniques learned by modelling spectroscopic data include the usefulness of mean-centring and, for regression problems, the use of the mean response value as initialisation for the weight in the ANN prediction layer. The last trick helps the network reach faster convergence of the ANN models by using starting values close to the typical response values.

The typical small number of samples involved in spectroscopic data poses challenges for DL modelling. Few training samples mean the DL model easily overfits the training data and causes a large variation in the estimation of the model performance. This makes hyperparameter tuning and model validation a frustrating aspect. Even two identical network architectures trained on the same data can have similar predictive performance but completely different model weights caused by different weight initialisation.

To picture this, one can look at two ANN models with the same architecture as the biological brains of two identical twins. At birth, each brain has the same structure but the initial strengths of the synapses (connections between the biological neurons) are different. This corresponds to different weight initialisation of the ANNs. During their lives, each twin gets the exact same sensory inputs but their brains develop differently because of the initial differences in the connections between the neurons. Despite their differences in neuron connections, the twin can make identical choices and behave the same in different situations. This is analogous to two ANNs giving the same predictions on the same data despite having very different model weights. When the twins have been subject to much sensory information (analogous to big data for ANNs), the two brains develop a kind of common understanding of the world and thus the two twins behave more similarly. On the other hand, if the twins receive less sensory information or just information of the same type, the two different brains are more likely to process other sensory information differently (new data), leading to more dissimilar behaviour of the twins. The different behaviours of the twins are an illustration of the high variance observed in ANN predictions when the models are trained on too little data.

4.2.2 Applying DL models in a traditional framework

Traditional and machine learning approaches work differently in the sense that machine learning models learn whatever is relevant to the problem automatically from the data without humans explicitly trying to incorporate knowledge into the model. It is hard to argue against the viability of the machine learning approach when one sees the achievements in image recognition tasks and large language modelling such as the recent ChatGPT model. However, one key factor for a successful DL model is to have plenty of data. Since this is not

easy to obtain in spectroscopy, traditional methods are often the better choice. As an alternative to pure DL modelling, Papers I and II explore different alternatives of using DL models in a more traditional framework. Paper I uses DL models to learn optimal preprocessing to be used on a traditional PLS model. Paper II utilises the flexibility in ANN architectures to model data in the more traditional framework of multiblock analysis and to quantify the importance of each data block. The pure DL alternative to modelling such multiblock data would be to just concatenate all features to one large data matrix and let the network do whatever it wants. This might lead to a more accurate model since the network has more freedom to connect the input features. However, it would be near impossible to say anything about feature importance in that case.

4.2.3 Improving traditional methods using DL techniques

The main research question in Paper III is to develop a framework where DL technology is used to improve the prediction performance of traditional methods while retaining interpretability. The proposed method is to simply use a linear model to capture all the linear relations between the features and the output and let a DL model figure out the non-linear relations from the linear model residuals. Although this modelling approach doesn't improve the linear model directly, the combined linear and DL prediction is a boost in prediction performance compared to the single linear model. This is a way to utilise the DL models without having to sacrifice model transparency since the linear model is still interpretable. An attempt to improve traditional preprocessing was done in Paper I, but the improvement was not large compared to classical preprocessing techniques and the interpretation of the DL-generated preprocessing turned out to be questionable.

4.2.4 Alternatives to DL models

After having explored how DL models can be used to model spectroscopic data both with and without the help of traditional approaches in Papers I and III and how DL models can be applied in the framework of traditional modelling in Papers I and II, Paper IV explores more traditional alternatives to DL models that are inspired by neural networks. DL models are powerful and useful in many situations. They are, however, not always applicable where little data is available and the model needs to be transparent such as in applications in health sciences. It is important to have alternatives that still are able to tackle complex problems. Paper IV addresses this by proposing

a flexible framework for using Locally Weighted Ensemble Models (LoWEM). Each model is locally linear but the combination makes it possible to model more complicated interactions both for regression and classification problems. There is a connection to ANNs in theories about the inner workings of ANNs as discussed in the paper. Furthermore, the framework is possible to scale up to really huge datasets by using a linear Perceptron as the class of local models as an alternative to an ordinary least squares (OLS) model. The Perceptron supports batch-wise training and can benefit from highly efficient implementations and GPU utilisation in software such as Tensorflow to tackle large problems. The LoWEM concept is demonstrated on smaller datasets in the paper, but the upscaling to large datasets will be interesting to study in more detail in the future.

4.2.5 Transparency VS performance

A common argument for the use of DL models is the performance in terms of prediction accuracy. On the other hand, arguments against them are usually based on the black box aspect of these models. In this thesis, there have been examples where DL models beat a traditional PLS model but with more sophisticated modelling such as two-level PLSR (Paper I and IV) or LoWEM (Paper IV), it is possible to outperform DL models. The strength of DL models in these situations is therefore that they provide good performance with minimal domain knowledge. Traditional models such as PLS are often preferred by chemometricians because of the possibility of interpretation with tools such as score- and loading plots. For these interpretations to be possible, or even accurate, domain knowledge is needed. Furthermore, domain knowledge is required in order to compose more sophisticated models that are able to beat DL models.

Interpretation of traditional models is not completely unproblematic. For example, if models are overfitted the interpretations might not be valid. Also, different people might interpret the same results differently. Even linear models such as PLS become more challenging to interpret when the number of components gets large and the cluster-based local modelling presented in Paper IV becomes confusing with many clusters. Another consideration is whether interpretability is always needed in a model. Accurate black box models can still be used to simplify certain tasks, but one has to be mindful of how these models are used and be critical when using them to make decisions.

DL models can to some degree be interpreted as seen in Paper II with respect to feature importance or one can study the feature maps of the model to try and explain parts of the model. In Paper I, an attempt to force the neural

network to do a specific preprocessing was made, but the DL model ended up doing something more complicated and non-transparent. A similar behaviour was experienced in Paper III when trying to force the DL model to learn the linear model residuals for classification. An ANN model is only concerned about minimising the objective function (loss function) it has been given and does this in the simplest possible ways it can find. This behaviour of the network caused at least two ideas to be scrapped from this project. At one point it was attempted to provide information about known subgroups in a dataset to the DL model by having two output nodes, one for predicting the known subgroups and one for predicting a continuous response value. It was, however, not possible to show any improvements in the network with this network configuration and it was hard to tell how the network managed the additional information it was given. The second failed attempt was an implementation of a PLS layer as a module for the neural network. The idea was to use the dimensionality reduction properties of the PLS model to improve the ANN weight updates in terms of efficiency and to more useful features for a PLS model. The resulting ANN model instead became unstable and did not show any promises. These two examples just illustrate the difficulties in trying to force a black box model which is not completely understood to do something specific.

4.3 Future perspectives

A challenge in spectroscopy is the small amount of available labelled data. One avenue of future work is to develop modelling strategies that can scale up to more data to get more accurate models. Paper IV is a step in that direction and does not rely on DL models. Further development into this framework to make the transition from iteration to iteration smoother is something to look into. Additionally, a faster implementation that utilises parallelisation would be useful.

The DL models already show promise in spectroscopy today, and it is reasonable to believe they will only get better with more data. Developing techniques to be able to train DL models on spectroscopic data from different sources is an interesting idea to increase the sample size. It could be possible to train a neural network to extract general features across the different spectroscopic sources, similar to ANN models used for transfer learning. Some early experimentation has shown that it is possible to build a network that has multiple input heads connected to a main body network and multiple outputs, one input-output pair for each data source. It remains to be seen if such a network architecture is able to work as the intended general feature extractor or if the ANN model decides to do something completely different again.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Chenhao Cui and Tom Fearn. Modern practical convolutional neural networks for multivariate regression: Applications to nir calibration. *Chemometrics and Intelligent Laboratory Systems*, 182:9–20, 2018.

[3] Jacopo Acquarelli, Twan van Laarhoven, Jan Gerretzen, Thanh N. Tran, Lutgarde M.C. Buydens, and Elena Marchiori. Convolutional neural networks for vibrational spectroscopic data analysis. *Analytica Chimica Acta*, 954:22–31, 2017.

[4] Salim Malek, Farid Melgani, and Yakoub Bazi. One-dimensional convolutional neural networks for spectroscopic signal regression. *Journal of Chemometrics*, 32(5):e2977, 2018.

[5] Tormod Næs, Knut Kvaal, Tomas Isaksson, and Charles Miller. Artificial neural networks in multivariate calibration. *Journal of Near Infrared Spectroscopy*, 1(1):1–11, 1993.

[6] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[8] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, 2010.

[9] Eirik Almklov Magnussen, Johanne Heitmann Solheim, Uladzislau Blazhko, Valeria Tafintseva, Kristin Tøndel, Kristian Hovde Liland, Simona Dzurendova, Volha Shapaval, Christophe Sandt, Ferenc Borondics, and Achim

Kohler. Deep convolutional neural network recovers pure absorbance spectra from highly scatter-distorted spectra of cells. *Journal of Biophotonics*, 13(12), 2020.

[10] Miroslav Kuchta, Sileshi Gizachew Wubshet, Nils Kristian Afseth, Kent-André Mardal, and Kristian Hovde Liland. Encoder–decoder neural networks for predicting future ftir spectra – application to enzymatic protein hydrolysis. *Journal of Biophotonics*, 15(9), 2022.

[11] Uladzislau Blazhko, Volha Shapaval, Vassili Kovalev, and Achim Kohler. Comparison of augmentation and pre-processing for deep learning and chemometric classification of infrared spectra. *Chemometrics and Intelligent Laboratory Systems*, 215:104367, 2021.

[12] Talha Naveed. Explore the effect of data augmentation of spectroscopic data for deep learning models. Accsessed: 07/04/2023.

[13] M. Mamouei, K. Budidha, N. Baishya, M. Qassem, and P. A. Kyriacou. An empirical investigation of deviations from the beer–lambert law in optical estimation of lactate. *Scientific Reports*, 11(1):13734, 2021.

[14] Roman Rosipal and Leonard J. Trejo. Kernel partial least squares regression in reproducing kernel hilbert space. *Journal of Machine Learning Research*, 2:97–123, 2001.

[15] Svante Wold, Harold Martens, and Herman Wold. The multivariate calibration problem in chemistry solved by the pls method. In *Matrix pencils*, pages 286–293, 1983.

[16] W. Fred McClure and Donald L. Stanfield. Near-infrared spectroscopy of biomaterials. In *Handbook of Vibrational Spectroscopy*. Wiley, 2006.

[17] Hicham Nocairi, El Mostafa Qannari, Evelyne Vigneau, and Dominique Bertrand. Discrimination on latent components with respect to patterns. application to multicollinear data. *Computational Statistics and Data Analysis*, 48(1):139–147, 2005.

[18] Åsmund Rinnan. Pre-processing in vibrational spectroscopy – when, why and how. *Anal. Methods*, 6(18):7124–7129, 2014.

[19] Harald Martens, SA Jensen, and P Geladi. Multivariate linearity transformation for near-infrared reflectance spectrometry. In *Proceedings of the Nordic symposium on applied statistics*, pages 205–234. Stokkand Forlag Publishers, 1983.

[20] Harald Martens and Edward Stark. Extended multiplicative signal correction and spectral interference subtraction: New preprocessing methods for near infrared spectroscopy. *Journal of Pharmaceutical and Biomedical Analysis*, 9(8):625–635, 1991.

[21] Nils Kristian Afseth and Achim Kohler. Extended multiplicative signal correction in vibrational spectroscopy, a tutorial. *Chemometrics and Intelligent Laboratory Systems*, 117:92–99, 2012.

[22] A. Kohler, U. Böcker, J. Warringer, A. Blomberg, S. W. Omholt, E. Stark, and H. Martens. Reducing inter-replicate variation in fourier transform infrared spectroscopy by extended multiplicative signal correction. *Applied Spectroscopy*, 63(3):296–305, 2009.

[23] Paul Bassan, Achim Kohler, Harald Martens, Joe Lee, Edward Jackson, Nicholas Lockyer, Paul Dumas, Michael Brown, Noel Clarke, and Peter Gardner. Rmies-emsc correction for infrared spectra of biological cells: Extension using full mie theory and gpu computing. *Journal of Biophotonics*, 3(8-9):609–620, 2010.

[24] R. J. Barnes, M. S. Dhanoa, and Susan J. Lister. Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra. *Applied Spectroscopy*, 43(5):772–777, 1989.

[25] Abraham. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[26] Kenneth Aase Kristoffersen, Kristian Hovde Liland, Ulrike Böcker, Sileshi Gizachew Wubshet, Diana Lindberg, Svein Jarle Horn, and Nils Kristian Afseth. Ftir-based hierarchical modeling for prediction of average molecular weights of protein hydrolysates. *Talanta*, 205:120084, 2019.

[27] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Plenum, 1981.

[28] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[29] Francois Chollet. *Deep Learning With Python*. Manning, 2 edition, 2021.

[30] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2015. arXiv: 1511.07289v5 [cs.LG].

[31] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[32] Alejandro F Frangi, Joachim Hornegger, Nassir Navab, and William M Wells. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing AG, Switzerland, 2015.

[33] Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search, 2019. arXiv: 1910.11858 [cs.LG].

[34] Jialin Dong, Mingjian Hong, Yi Xu, and Xiangquan Zheng. A practical convolutional neural network model for discriminating raman spectra of human and animal blood. *Journal of Chemometrics*, 33(11), 2019.

[35] Randall Balestriero and richard baraniuk. A spline theory of deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 374–383. PMLR, 2018.

Paper I

Received: 9 March 2021 | Revised: 3 September 2021 | Accepted: 6 September 2021

DOI: 10.1002/cem.3374

SPECIAL ISSUE - RESEARCH ARTICLE

Journal of CHEMOMETRICS WILEY

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

1 | BACKGROUND

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.[1] Traditional analysis methods include principal component analysis (PCA),[2] partial least squares (PLS) regression[3,4] and support vector machines (SVMs).[5] 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[6] in 1958. Important milestones in training and designing network architectures include the back-propagation algorithm[7,8] invented in 1986, and the advent of convolutional neural networks (CNNs) in 1989.[9,10] Further development led to the artificial intelligence (AI) revolution in image-based classification and descriptions, starting with the AlexNet[11] 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 $\mathbf{w}$ (*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,[12] also known as the Bernoulli log-likelihood loss: $\sum_{i}^{n}-\mathbf{y}_i\log(\hat{\mathbf{y}}_i)$ for classification problems, where $\mathbf{y}_i$ is a one-hot (dummy) encoded class label vector and $\hat{\mathbf{y}}_i$ 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

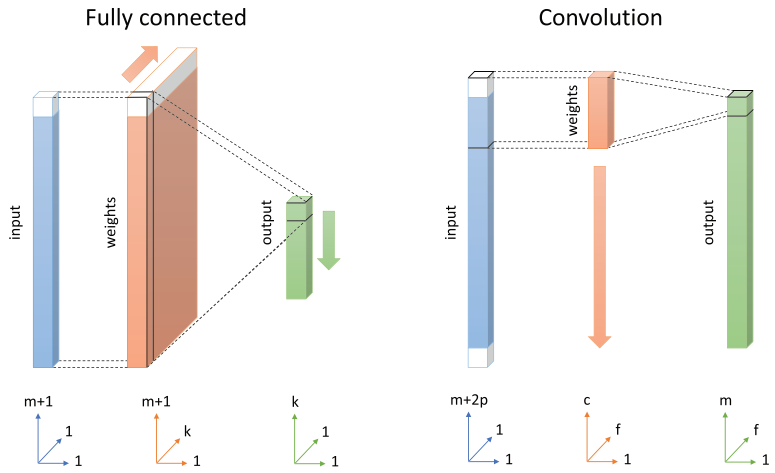


FIGURE 1 Illustrations of a dense and convolutional connection in a 1D artificial neural network (ANN). The fully connected layer to the left contains k hidden units (neurons), where the white parts of the input and weights represent the bias units. The illustrated convolutional layer to the right contains f filters, each including c weights. The white parts of the convolutional neural network (CNN) input layer represents padding with zeros

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

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.[13] For more details regarding CNNs, see Goodfellow.[14]

The use of ANNs with spectroscopic data was explored by Næs et al[15] with an emphasis on near infrared (NIR) applications. More recent studies that utilise CNNs include Liu et al,[16] 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),[17] SVMs[5] and random forests.[18] Acquarelli et al[19] 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[20] achieved good performance using CNN on NIR calibration, and Malek et al[21] 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).[22] 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).[23] 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[24] and the SciPy[25] 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)[26] 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 ($\boldsymbol{X}_{ref}$) and additional terms corresponding to the polynomial trends of degree $i = 1, \ldots, n$ ($\boldsymbol{\nu}^i$) plus a residual term ($\boldsymbol{e}$):

$$\boldsymbol{X} = a + \boldsymbol{X}_{ref} \cdot b + \sum_{i=1}^{n} d_i \boldsymbol{\nu}^i + \boldsymbol{e}.$$

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, $\boldsymbol{e}$. The EMSC-corrected spectrum can then be expressed as follows:

$$\boldsymbol{X}_{corr} = \frac{\boldsymbol{X} - a - \sum_{i=1}^{n} d_i \boldsymbol{\nu}^i}{b} = \boldsymbol{X}_{ref} + \frac{\boldsymbol{e}}{b}. \quad (1)$$

Direct extensions of the EMSC method handle interferents,[27] replicate variation,[28] Mie scattering,[29] multiple references[30] and more. The EMSC method is useful for a range of different kinds of spectroscopic techniques, such as NIR,[22] Raman[31] and FTIR.[27]

We also note the close relationship between the standard normal variate (SNV) transformation and the MSC.[32] If the coefficients a and b in Equation (1) represent the mean and standard deviation of the spectrum $\boldsymbol{X}$, 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[33] 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 $\boldsymbol{w} = [w_1, w_2, \ldots, w_k]^T$ (where k is the filter size), which sum to 1:

$$w_i \geq 0, \sum_{i=1}^{k} w_i = 1, \text{ for } i = 1, 2, \ldots, k. \tag{2}$$

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:

$$\boldsymbol{X}_{corrected} = \boldsymbol{X} - \boldsymbol{X}_{smoothed} = \boldsymbol{X} - \boldsymbol{X} * h(\cdot) = \boldsymbol{X} * (\boldsymbol{I} - h(\cdot)), \tag{3}$$

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

$$w_i \leq I_i, \sum_{i=1}^{k} w_i = 0, \text{ for } i = 1, 2, \ldots, k, \tag{4}$$

that is, the weights must sum to 0, and each weight must be smaller than the identity kernel $\boldsymbol{I}$. See Dong et al[33] 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.[34] 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[3,4] for our comparisons. We refer to Wold et al[35,36] 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,[37] 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[33] 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[38] 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.[39] 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[39] 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.

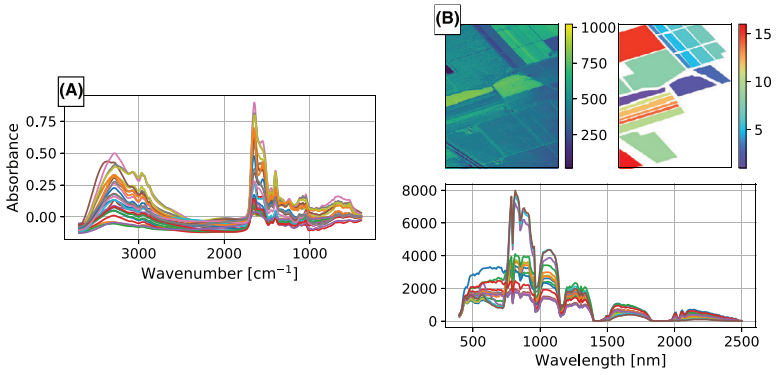


FIGURE 2 Samples from the two datasets. (A) One spectrum for each enzyme-material class in the Fourier-transform infrared (FTIR) dataset before preprocessing. The spectral range depicted is the same as passed to the models in the analysis. (B) Remote-sensing dataset. Top row: a sample image of one spectral 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 × 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).

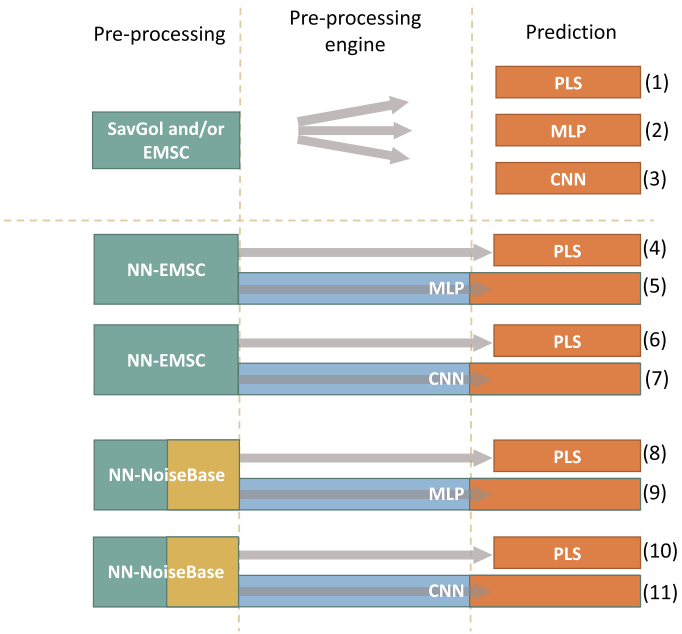


FIGURE 3 Overview of the various pipelines considered. "SavGol" is short for Savitzky–Golay filtering. The two colors of "NN-NoiseBase" indicate that the method consists of two parts: denoising (green) and baseline correction (yellow)

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.

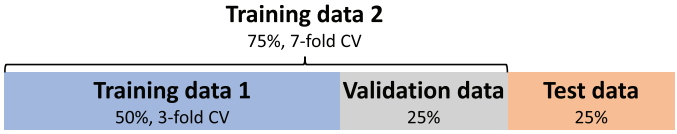


FIGURE 4 Data splitting scheme for the two analysis phases. "Training data 1" and "Validation data" are used for parameter selection. "Training data 2" and "Test data" are used for model selection. Note that "Train data 2" = "Train data 1" ∪ "Validation data"

TABLE 1 Preprocessing hyperparameters

Preprocessing method	Hyperparameters
Savitzky–Golay	Window size: {7, 9, 11, 13}
	Derivative: {0, 1, 2}
	Poly. degree: {2, 3}
EMSC	Poly. degree: {0, 1, 2}
NN-NoiseBase	Denoising filter (both datasets): {3, 5, ..., 19}
	Baseline correction filter (FTIR): {11, 36, ..., 211}
	Baseline correction filter (remote-sensing): {11, 21, ..., 91}
NN-EMSC	Poly. degree: {0, 1, 2}

Abbreviations: EMSC, extended multiplicative signal correction; FTIR, Fourier-transform infrared; NN-EMSC, neural network extended multiplicative signal correction; NN-NoiseBase, neural network denoising and baseline correction.

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.

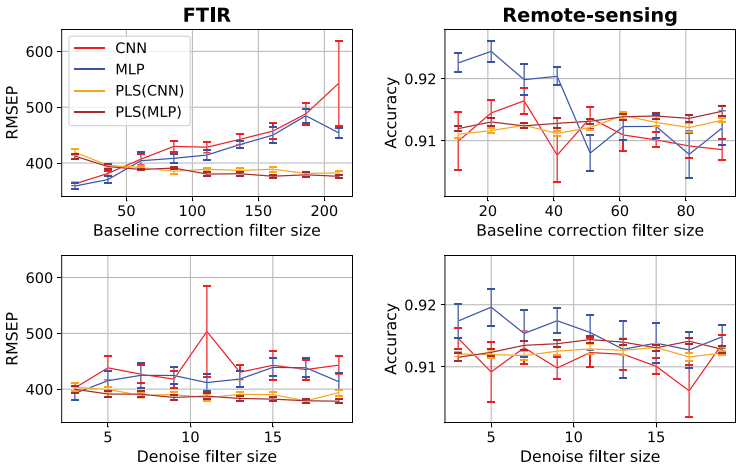


FIGURE 5 The prediction results on the validation set during the parameter selection phase of the pipelines using the NN-NoiseBase preprocessing technique for different sizes of the denoising and baseline-correction filters. Left column: Fourier-transform infrared (FTIR) dataset. Right column: remote-sensing 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	512	465.0 ± 3.01	0.910	0.91 ± 0.0016
Raw		CNN	426	353.0 ± 7.0	0.925	0.926 ± 0.0012
Raw	-	PLS	457	426.0 ± 0.0	0.913	0.914 ± 0.0
SG-EMSC		MLP	439	357.0 ± 2.57	0.949	**0.948 ± 0.0007**
SG-EMSC		CNN	401	323.0 ± 3.96	0.950	**0.948 ± 0.001**
SG-EMSC	-	PLS	457	415.0 ± 0.0	0.917	0.918 ± 0.0
NN-EMSC		MLP	543	514.0 ± 5.92	0.929	0.929 ± 0.001
NN-EMSC		CNN	427	319.0 ± 6.93	0.928	0.922 ± 0.0017
NN-EMSC	MLP	PLS	442	421.0 ± 0.35	0.915	0.915 ± 0.0001
NN-EMSC	CNN	PLS	420	406.0 ± 1.57	0.916	0.916 ± 0.0001
NN-NoiseBase		MLP	427	**316.0 ± 4.34**	0.939	0.938 ± 0.0029
NN-NoiseBase		CNN	425	336.0 ± 4.2	0.934	0.926 ± 0.0017
NN-NoiseBase	MLP	PLS	450	413.0 ± 1.1	0.916	0.912 ± 0.0002
NN-NoiseBase	CNN	PLS	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

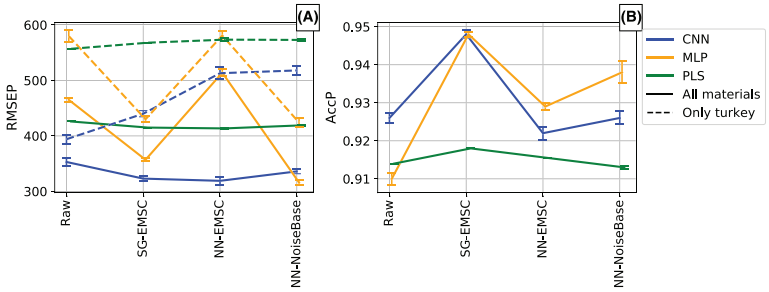


FIGURE 6 Results from the model selection phase. (A) Results for the Fourier-transform infrared (FTIR) dataset. Solid lines: mean root mean squared error of predictions (RMSEP) over 30 repetitions with standard error. Dotted lines: mean RMSEP and standard error of predictions on the test samples containing turkey. (B) The results from the remote-sensing dataset. The lines show the mean accuracy predicted (AccP) over 30 repetitions with standard error. The partial least squares (PLS) performances for the neural network extended multiplicative signal correction (NN-EMSC) and neural network denoising and baseline correction (NN-NoiseBase) techniques are averaged over the two preprocessing engines for both datasets

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 th 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

TABLE 3 Time usage for training of each prediction model and the additional time of including the preprocessing alternatives (in seconds)

Classical alternatives	FTIR	Remote-sensing
PLS/PLS-DA	1.0	15.4
SavGol	+0.011	+0.022
EMSC	+0.024	+0.027
DL alternatives	**FTIR**	**Remote-sensing**
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.

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.

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.[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 preprediction layers, it is harder to discern the exact roles of the different layers as 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 loosing 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[39] 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.[39] 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 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 straight forward 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[40] 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 J, 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. Köhler 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, Wubshet 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*. 2021;e3374. doi:10.1002/cem.3374

Paper II

Ranking Feature-Block Importance in Artificial Multiblock Neural Networks

Anna Jenul( ), Stefan Schrunner, Bao Ngoc Huynh, Runar Helin, Cecilia Marie Futsæther, Kristian Hovde Liland, and Oliver Tomic

Norwegian University of Life Sciences, Universitetstunet 3, 1432 Ås, Norway
{anna.jenul,stefan.schrunner,ngoc.huynh.bao,runar.helin,
cecilia.futsaether,kristian.liland,oliver.tomic}@nmbu.no

Abstract. In artificial neural networks, understanding the contributions of input features on the prediction fosters model explainability and delivers relevant information about the dataset. While typical setups for feature importance ranking assess input features individually, in this study, we go one step further and rank the importance of groups of features, denoted as feature-blocks. A feature-block can contain features of a specific type or features derived from a particular source, which are presented to the neural network in separate input branches (multiblock ANNs). This work presents three methods pursuing distinct strategies to rank feature-blocks in multiblock ANNs by their importance: (1) a composite strategy building on individual feature importance rankings, (2) a knock-in, and (3) a knock-out strategy. While the composite strategy builds on state-of-the-art feature importance rankings, knock-in and knock-out strategies evaluate the block as a whole via a mutual information criterion. Our experiments consist of a simulation study validating all three approaches, followed by a case study on two distinct real-world datasets to compare the strategies. We conclude that each strategy has its merits for specific application scenarios.

Keywords: Feature-block importance · Importance ranking · Multiblock neural network · Explainability · Mutual information

1 Introduction

In machine learning, datasets with an intrinsic block-wise input structure are common; blocks may represent distinct data sources or features of different types and are frequently present in datasets from industry [7], biology [3], or healthcare [5]. For example, in healthcare, heterogeneous data blocks like patient histology, genetics, clinical data, and image data are combined in outcome prediction models. However, good prediction models do not necessarily depend equally on each block. Instead, some blocks may be redundant or non-informative. Identifying the key data sources in multi-source treatment outcome models promises to deliver new insights into the behavior of black-box models like ANNs. In particular, potential benefits include improving model explainability, reducing costly

E. Pimenidis et al. (Eds.): ICANN 2022, LNCS 13532, pp. 163–175, 2022.
https://doi.org/10.1007/978-3-031-15937-4_14

data acquisitions that do not contribute to the model prediction, and allowing domain experts to explore latent relations in the data. Thus, there is a need to measure the importances of feature-blocks, denoted as feature-block importance ranking (BIR).

In order to exploit the internal structure of the block-wise data in neural networks, a multiblock ANN (M-ANN) architecture is used. As depicted in Fig. 1, the M-ANN consists of a separate input branch for each block, a concatenation layer to merge information from all branches, and a blender network to map the information to the model output. The architecture allows for any type of network layer, depth, activation, or other network parameters, including the special case where the concatenation layer equals the input layer (block branches of depth 0).

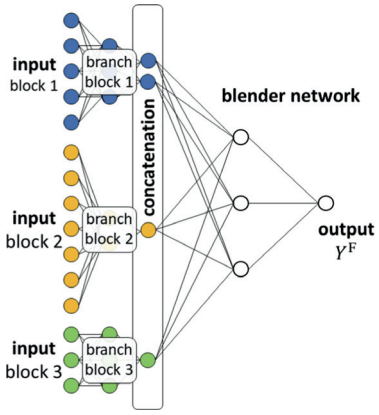


Fig. 1. M-ANN architecture.

Ranking individual features by their importances (feature importance ranking, FIR) has been studied for different types of ANNs [8,14,15]. An extensive evaluation [9] showed that versions of the variational gradient method (VarGrad) [1,2] outperformed competitors such as guided backprop and integrated gradients. For BIR, however, a combination of features in one block may accumulate a larger amount of information than each feature separately due to informative non-linear relations between features. Hence, using FIR might oversimplify the problem of measuring block importance since interactions between features of the same block are disregarded. Nevertheless, the strategy of reducing BIR to a simple summary metric (sum, mean, max) over FIR scores is considered in our evaluation.

A related problem to FIR is feature selection, where the input dimensionality is reduced to the most influential features as part of the preprocessing. Feature selection is widely studied in ANNs. Furthermore, specialized feature selectors can account for block structures like UBayFS [11] or groupLasso [16]. Conceptually, these feature selectors aim to improve model performance and classify an entire block as important/unimportant in a binary way before model training. In contrast, our BIR problem is considered a post-processing procedure, focusing on analyzing the model after training without influence on the model performance.[1]

This study presents and discusses three distinct approaches to quantify the importance of feature-blocks in M-ANNs. While exploiting the flexibility of ANNs and their capacities to learn complex underlying patterns in the data, the

[1] BIR may be used for block feature selection if deployed as filter method—however, this aspect is beyond the scope of the present work.

discussed methods aim to deliver insights into the trained network's dependence structure on the distinct input blocks and thereby foster model explainability. We propose three paradigms for BIR: a block is considered as important if

1. it consists of features with high FIR scores (composition strategy), or
2. it explains a large part of the network output (knock-in strategy), or
3. its removal significantly changes the network output (knock-out strategy).

We evaluate and discuss the proposed paradigms in a simulation study and present two case studies on real-world datasets, where the behaviors of the proposed ranking strategies become apparent.

In the following, bold letters denote vectors and matrices; non-bold letters denote scalars, functions or sets. Square brackets denote index sets $[n] = \{1, \ldots, n\}$.

2 Block Importance Ranking Methods

We assume data input $\boldsymbol{x}$ from some feature space $D \subset \mathbb{R}^N$, $N \in \mathbb{N}$, following a probability distribution $\boldsymbol{X} \sim P_{\boldsymbol{X}}$, and a univariate target variable $y \in T \subset \mathbb{R}$ following a probability distribution $Y \sim P_Y$. Given training data $(\boldsymbol{x}, y) \in D_{\text{train}} \times T_{\text{train}} \subset D \times T$, model parameters $\boldsymbol{w} \in W \subset \mathbb{R}^M$, $M \in \mathbb{N}$, are trained with respect to some loss term $e : D \times T \to \mathbb{R}^+$,

$$\boldsymbol{w}^\star = \min_{\boldsymbol{w} \in W} e\left(f_{\boldsymbol{w}}(\boldsymbol{x}), y\right),$$

where the ANN is a function $f_{\boldsymbol{w}} : D \to T$ given weights $\boldsymbol{w}$.

In an M-ANN architecture, see Fig. 1, the block structure of the model input is represented by a direct sum of subspaces $D = \bigoplus_{b=1}^{B} D_b$, each corresponding to one block $b \in [B]$ with dimension $N_b = \dim(D_b)$, $N = \sum_{b=1}^{B} N_b$. Each block enters a distinct branch of the network that processes the block input. Afterwards, the outputs of all branches are merged in a concatenation layer, which consists of n_b nodes associated with each block b, respectively. A so-called blender network $f_{\boldsymbol{w}}^{\text{blender}}$ connects the concatenation layer to the network output. Network training is performed using backpropagation, where all block branches and the blender network are trained simultaneously in an end-to-end manner.

2.1 Composite Strategy

Our first paradigm composes block importance measures from FIR in a direct way. As a prototype of state-of-the-art FIR methods, we use VarGrad [2]. VarGrad builds on the idea that variations of an important feature provoke measurable variations in the output. Under the assumption that features are on a common scale, we estimate the gradient of the function $f_{\boldsymbol{w}}$ with respect to each feature by adding small random perturbations in the input layer. A large

variance in the gradient indicates that the network output depends strongly on a feature, i.e., the feature is important. We denote the importance of feature $n \in [N]$ as quantified by VarGrad, by $\alpha_n \in \mathbb{R}^+$.

To translate the feature-wise importance measure to feature-blocks in M-ANNs, we deploy a summary metric φ over all single-feature importances in a block $b \in [B]$. Thus, block importances $\gamma_\varphi^{(b)}$ are defined as

$$\gamma_\varphi^{(b)} = \varphi(\alpha_1^{(b)}, \ldots, \alpha_{N_b}^{(b)}), \tag{1}$$

where $\alpha_n^{(b)}$ denotes the n^{th} feature associated with the b^{th} block. Intuitive choices for φ are either the sum, mean, or maximum operator, denoted as φ_{sum}, φ_{mean}, or φ_{max}, respectively. Rankings based on mean and sum are equal, if all blocks contain the same number of features. Operators φ_{sum} and φ_{mean} accumulate the individual feature importances: a block with multiple features of high average importances is preferred over blocks with few top features and numerous unimportant features. In contrast, φ_{max} compares the top-performing features out of each block, while neglecting all other's contributions. Statistical properties of block importance quantifiers implementing the composite strategy are transmitted from (i) the feature importance ranking method and (ii) the summary metric. Since this approach cannot capture between-feature relations, potentially impacting the importance of a block, we suggest two other paradigms.

2.2 Knock-In Strategy

The knock-in strategy is inspired by work on the information bottleneck [4], demonstrating that node activations can be exploited for model interpretation in ANNs. In the concatenation layer of the M-ANN (Fig. 1), where information from the blocks enters the blender network, activations are of particular importance since they represent an encoding of the block information. When passing model input $\boldsymbol{x}$ through the network, we denote the activation of the nth node associated with block $b \in [B]$ in the concatenation layer by $c_{b,n}(\boldsymbol{x})$, $n \in [n_b]$. The average activation of the nth node in block $b \in [B]$ across all training data $\boldsymbol{x} \in D_{\text{train}}$ is denoted by $\bar{c}_{b,n}$.

For BIR, we compute a pseudo-output by passing data of only one block b through the network. For this purpose, we introduce a pseudo-input $\boldsymbol{v}^{(b)}(\boldsymbol{x})$ as

$$v_{b',n}^{(b)}(\boldsymbol{x}) = \begin{cases} c_{b',n}(\boldsymbol{x}) & \text{if } b' = b \\ \bar{c}_{b',n} & \text{otherwise,} \end{cases} \tag{2}$$

where $b' \in [B]$, and $n \in [n_b]$. By propagating pseudo-input $\boldsymbol{v}^{(b)}(\boldsymbol{x})$ through the blender network, we obtain the pseudo-output $f_{\boldsymbol{w}}^{\text{blender}}(\boldsymbol{v}^{(b)}(\boldsymbol{x}))$. The main assumption behind the knock-in strategy is that high agreement between output $f_{\boldsymbol{w}}(\boldsymbol{x})$ and pseudo-output $f_{\boldsymbol{w}}^{\text{blender}}(\boldsymbol{v}^{(b)}(\boldsymbol{x}))$ indicates a high importance of block b, since information from b is sufficient to recover most of the model output. In contrast, a large discrepancy between the two quantities indicates low explanatory power of the block b, and thus, a lower block importance. The concept to generate knock-in pseudo-outputs is illustrated in Fig. 2.

We implement the knock-in concept via the mutual information (MI) [6], an information-theoretic measure to quantify the level of joint information between two discrete random variables Z and Z', defined as

$$\mathrm{MI}(Z, Z') = \sum_{z} \sum_{z'} p_{Z,Z'}(z, z') \log_2 \left(\frac{p_{Z,Z'}(z, z')}{p_Z(z) p_{Z'}(z')} \right).$$

If Z and Z' are independent, $\mathrm{MI}(Z, Z')$ is 0. Otherwise, $\mathrm{MI}(Z, Z')$ is positive, where a high value indicates a large overlap in information. To quantify the joint and marginal distributions of continuous variables Z and Z', two-dimensional and one-dimensional histograms can be used as non-parametric estimators for $p_{Z,Z'}$, p_Z, and $p_{Z'}$, respectively. We denote the number of equidistant histogram bins along each axis by $\ell \in \mathbb{N}$. It follows from the properties of entropy [6] that an upper bound to $\mathrm{MI}(Z, Z')$ is given by $\log_2(\ell)$.

As shown in Fig. 2, the random variable of (full) model output, $Y^{\mathrm{F}} = f_{\boldsymbol{w}}(\boldsymbol{X})$, and the random variable of the pseudo-output with respect to block b, $Y^{(b)} = f_{\boldsymbol{w}}^{\mathrm{blender}}(\boldsymbol{v}^{(b)}(\boldsymbol{X}))$, where $\boldsymbol{X}$ follows the input distribution $P_{\boldsymbol{X}}$, are used to measure knock-in (KI) block importance as

$$\gamma_{\mathrm{KI}}^{(b)} = \frac{\mathrm{MI}(Y^{\mathrm{F}}, Y^{(b)})}{\log_2(\ell)}. \tag{3}$$

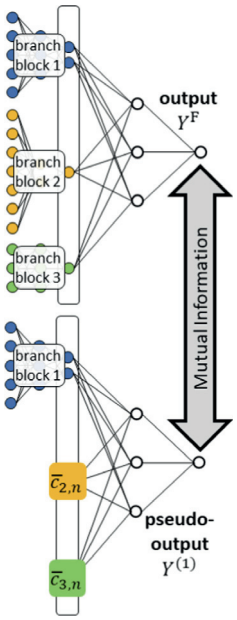


Fig. 2. Knock-in strategy: Pseudo-outputs for feature-block $b = 1$ are generated by activating block b, while imputing averaged activations for all other blocks.

2.3 Knock-Out Strategy

The knock-out paradigm is an ablation procedure where one block at a time is removed from the model in order to measure the impact of the remaining blocks. We pursue a similar approach as in the knock-in paradigm and specify knock-out pseudo-inputs $\boldsymbol{v}^{(-b)}(\boldsymbol{x})$ as

$$v_{b',n}^{(-b)}(\boldsymbol{x}) = \begin{cases} \bar{c}_{b',n} & \text{if } b' = b \\ c_{b',n}(\boldsymbol{x}) & \text{otherwise,} \end{cases} \tag{4}$$

for an arbitrary block $b \in [B]$. Thus, the definition in Eq. 4 represents an opposite behavior of Eq. 2 in the knock-in case. In analogy to the knock-in notation, we denote the random variable of pseudo-outputs with respect to $\boldsymbol{v}^{(-b)}$ as $Y^{(-b)} = f_{\boldsymbol{w}}^{\mathrm{blender}}(\boldsymbol{v}^{(-b)}(\boldsymbol{X}))$. The knock-out concept is illustrated in Fig. 3. In contrast to knock-in, we assume that leaving out block b having a relevant impact on the final output delivers a more dissimilar pseudo-output to the full output since relevant information is lost. Removing an unimportant block preserves the relevant information and delivers a pseudo-output similar to the full output.

Finally, we define the importance of block $b \in [B]$ with respect to the knock-out strategy (KO) as

$$\gamma_{\text{KO}}^{(b)} = \frac{\log_2(\ell) - \text{MI}(Y^{\text{F}}, Y^{(-b)})}{\log_2(\ell)}. \tag{5}$$

For both, KI and KO, importance scores $\gamma_{\text{KI}}^{(b)}$ and $\gamma_{\text{KO}}^{(b)}$ are bounded between 0 (unimportant block) and 1 (important block).

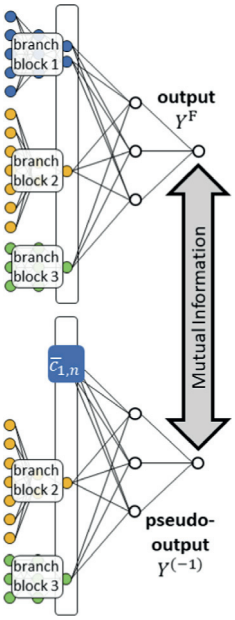


Fig. 3. Knock-out strategy: Pseudo-outputs are generated by activating all but one blocks.

3 Experiments

As a proof of concept, we conduct two experiments to assess BIR in M-ANNs. The first experiment involves six simulated, non-linear regression problems, where our simulation setup delivers information on the ground truth block importances. This experiment verifies that our suggested measures can identify the ground truth block rankings, defined by their corresponding paradigms. Real-world datasets are evaluated in two case studies in experiment 2, where no exact ground truth block ranking is available. Instead, we compare BIR strategies to each other.

3.1 Simulation Experiment

We simulate a synthetic datasets along with six distinct target functions, denoted as setups S1a–S1c and S2a–S2c. The dataset consists of $N = 256$ features, divided randomly into $B = 8$ blocks (B1–B8) à $N_b = 32$ features. The sample size is set to $|D_{\text{train}}| = 10\,000$ and $|D_{\text{test}}| = 10\,000$. All features are simulated from a multivariate normal distribution with mean vector $\boldsymbol{\mu} = \mathbf{0}$ and a randomized covariance matrix $\boldsymbol{\Sigma}$; hence a non-trivial correlation structure is imposed.[2]

Setups S1a–S1c and S2a–S2c differ in the parameters used to compute the non-linear target variable y, which is simulated via a noisy linear combination of the squared features with coefficient matrix $\boldsymbol{\beta}^{(b)} \in \mathbb{R}^{N_b \times N_b}$, given as

$$y = \underbrace{\sum_{b=1}^{8} \boldsymbol{x}^T \boldsymbol{\beta}^{(b)} \boldsymbol{x}}_{g(\boldsymbol{x})} + \varepsilon_{\text{noise}}, \text{ where } \quad \boldsymbol{\beta}^{(b)} = \left(\begin{array}{ccc|cc} \beta_{\text{imp}} & 0 & \vdots & 0 & 0 \;\vdots\; 0 \\ \beta_{\text{int}} & \beta_{\text{imp}} & \vdots & 0 & 0 \;\vdots\; 0 \\ \cdots & & \ddots & & \\ \beta_{\text{int}} & \beta_{\text{int}} & & \beta_{\text{imp}} & 0 \quad 0 \\ \hline 0 & 0 & & 0 & 0 \quad 0 \\ & \cdots & & & \ddots \\ 0 & 0 & & 0 & 0 \quad 0 \end{array}\right),$$

$$\varepsilon_{\text{noise}} \underset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0, \sigma^2_{\text{noise}}\right), \text{ and} \tag{6}$$

[2] Code and details on simulation and network architecture are available at https://github.com/annajenul/Block_Importance_Quantification.

Table 1. Specifications for matrix $\boldsymbol{\beta}^{(b)}$: block importance is steered via count N_{imp}, coefficient β_{imp}, and interaction β_{int} of the important features.

Setup	Block												
	B1			B2			...	B7			B8		
	N_{imp}	β_{imp}	β_{int}	N_{imp}	β_{imp}	β_{int}	...	N_{imp}	β_{imp}	β_{int}	N_{imp}	β_{imp}	β_{int}
S1a	2	7	0	2	6	0	...	2	1	0	0	0	0
S1b	7	2	0	6	2	0	...	1	2	0	0	0	0
S1c	1	7	0	2	6	0	...	7	1	0	0	0	0
S2a	2	7	1	2	6	1	...	2	1	1	0	0	1
S2b	7	2	1	6	2	1	...	1	2	1	0	0	1
S2c	1	7	1	2	6	1	...	7	1	1	0	0	1

The matrix $\boldsymbol{\beta}^{(b)} \in \mathbb{R}^{N_b \times N_b}$ contains an $N_{\text{imp}} \times N_{\text{imp}}$ quadratic sub-matrix consisting of coefficients β_{imp} of important features, i.e. features with relevant contribution to the target, and interactions β_{int}. The noise parameter σ_{noise} is set to 10% of the standard deviation of the linear combination $g(\boldsymbol{x})$ across the generated samples $\boldsymbol{x}$. As shown in Table 1, block importances are varied between the setups and as follows

- S1a: varying coefficients of important features, but constant counts;
- S1b: varying counts of important features, but constant coefficients;
- S1c: varying counts and coefficients of important features;
- S2a–S2c: same as S1a–S1c, but with interaction terms between features.

Due to the randomized correlation matrix of the feature generation, unimportant features may be correlated with important features, as well as with the target y.

For each setup, we trained the described M-ANN model in 30 independent runs with distinct weight initializations after data standardization. Since BIR methods are deployed post-hoc and assume a model with appropriate performance, runs with poor performances (R2 < 0.8) were excluded from the analysis after outlier removal. Hence, the number of model runs in the analysis was 20 (S1a, S1b, S2a, S2b), 18 (S1c), and 19 (S2c), respectively. The remaining models achieved an average performance of $\geq$0.9 (R2 score) and $\leq$0.2 (RMSEIQR: root mean squared error scaled by inter-quartile range) on the test set.

For evaluation, importance scores across all model runs were tested for significant differences using a pairwise Wilcoxon-test with Bonferroni correction. If the p-value in a comparison between two blocks was above a significance level of 0.01, both were counted as tie. Figure 4 illustrates the distributions of BIR scores after min-max-normalization by setup and method, along with rankings (colors) based on significant group differences. All methods discovered the intrinsic ranking in dataset S1a. In dataset S1b, knock-in, knock-out, and VarGrad-mean identified the ranking by underlying important feature counts N_{imp}, while VarGrad-max failed to deliver a significant distinction between blocks with higher counts of

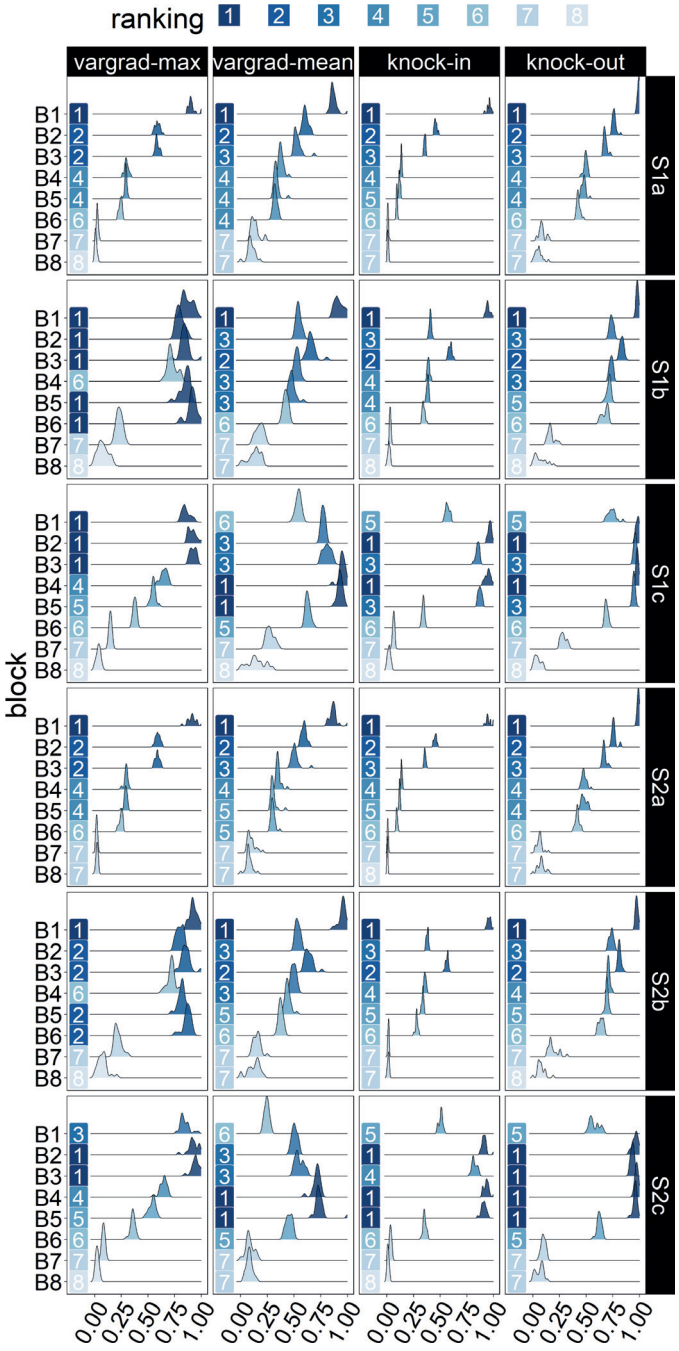


Fig. 4. Distributions of the normalized BIR scores across model runs. Rankings are indicated by colors and refer to significant group differences based on a pairwise Wilcoxon-test (significance level 0.01).

Table 2. Averaged Spearman's rank correlation coefficients comparing each ranking to the ground truth BIR for each paradigm across model runs. Standard deviations were $\leq$0.03 for S1a, S1b, S2a and S2b, and $\leq$0.06 for S1c and S2c.

Paradigm	Dataset					
	S1a	S1b	S1c	S2a	S2b	S2c
Composite (VarGrad-max)	0.97	0.58	0.93	0.98	0.58	0.91
Composite (VarGrad-mean)	0.98	0.95	0.96	0.97	0.95	−0.40
Knock-in	0.99	0.98	0.85	0.97	0.99	0.89
Knock-out	0.99	0.98	0.81	0.99	0.99	0.89

important features. For dataset S1c, VarGrad-max mostly ranked by underlying β_{imp} and ignored N_{imp}, while knock-in, knock-out and VarGrad-mean delivered trade-offs between counts N_{imp} and coefficients β_{imp} of important features. In setups S2a, S2b, and S2c with between-feature interactions, the same rankings as in S1a, S1b, and S1c could be obtained by all methods with negligible deviations. Hence, we conclude that all metrics remain stable in more complex scenarios.

We further validated the paradigms by comparing the results to their corresponding ground truth block importances, determined by the real coefficients in the simulation setup. For the composite max and mean paradigms, the corresponding maxima and means over $\boldsymbol{\beta}^{(b)}$, were used as references. Ground truth importances for knock-in (KI), and knock-out (KO) were based on the explained variances of the single block b in the underlying linear combination, given as

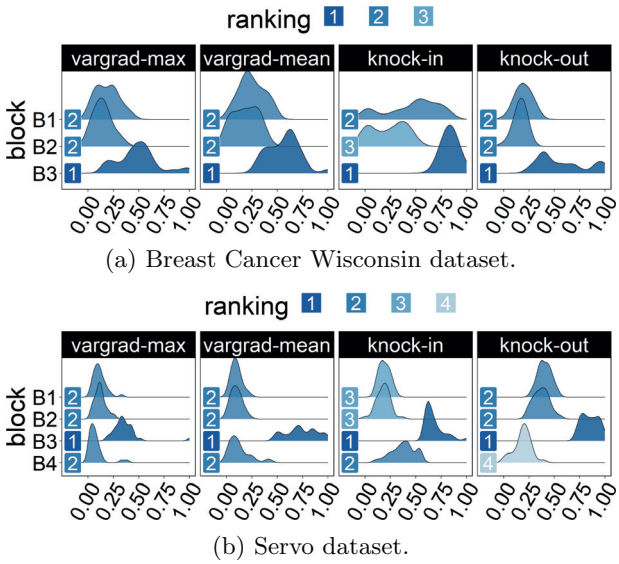
$$KI_b = \mathbb{E}\left(y - \left(\boldsymbol{x}^{(b)}\right)^T \boldsymbol{\beta}^{(b)}\boldsymbol{x}^{(b)}\right), \text{ and } KO_b = \mathbb{E}\left(y - \sum_{\substack{b'=1 \\ b'\neq b}}^{8} \left(\boldsymbol{x}^{(b')}\right)^T \boldsymbol{\beta}^{(b')}\boldsymbol{x}^{(b')}\right),$$

where $\boldsymbol{x}^{(b)}$ denotes projection of input $\boldsymbol{x}$ on the subspace of block b, D_b. The comparison between the rankings based on (average) predicted importance scores and ground truth rankings was made using Spearman's correlation coefficient, see Table 2. With two exceptions, all correlation values were at a high level, indicating that our methods accurately predicted the ground truth. Spearman's correlation coefficient is not representative in S1b, and S2b with respect to the maximum metric since the ground truth ranking is equal for blocks B1–B7. In S2c VarGrad-mean is distracted by decreasing β_{imp} and an increasing number of interaction terms, although underlying block importances are in increasing order with respect to the mean metric.

3.2 Real-World Experiment

Since verification on simulated data showed that the presented approaches match the ground truth according to their paradigms, we deployed the methods on two

real-world datasets, where underlying block importance is unknown. Prior to analysis, both datasets were standardized on the trained data. Again, we trained 30 independent model runs.



(a) Breast Cancer Wisconsin dataset.

(b) Servo dataset.

Fig. 5. Distributions of normalized BIR scores in experiment 2 across model runs.

The Breast Cancer Wisconsin dataset (BCW) [13] describes a binary classification problem (malignant or benign tumor) and consists of 569 samples (398 train, 171 test) and three blocks with ten features each, representing groups of distinct feature characteristics (mean values, standard deviations, and extreme values of measured parameters). The average performance was 0.95 (accuracy) and 0.96 (F1 score) without outliers. The average scores and rankings delivered by BIR methods are shown in Fig. 5a. All four paradigms discovered that block 3 is dominant, which agrees with previous research on the dataset [10]. However, knock-in was the only method that distinguished between the importances of B1 and B2. According to [10], block B1 contains overlapping information with B3, while B2 is rather non-informative. Thus, the experiment underlines a difference between knock-in and knock-out rankings in the presence of redundancies.

Servo [12] is a dataset containing 167 samples (120 train, 47 test), a univariate, numeric target variable, and four features, two of which are categorical variables with four levels each, and two are numerical variables. Each feature was assigned its own block. One-hot encoding was performed for the two blocks containing categorical features, leading to two blocks (B1 and B2) of four binary features, each. Blocks corresponding to numerical features (B3 and B4) contain one feature each. In the 30 M-ANN model runs, an average performance of 0.21 (RMSEIQR) and 0.87 (R2) was obtained without outliers. Figure 5b shows that

for all four paradigms, block B3 was most important. While VarGrad methods delivered a binary ranking, knock-in and knock-out suggested a ranking with 3 and 4 distinct importance levels, respectively—thus, the level of detail was higher in the MI-based rankings compared to VarGrad methods.

4 Discussion

Our experiments demonstrated several differences between the proposed strategies. While the composite strategy evaluates features individually and depends on two user-selected parameters (the feature-wise ranking scheme and the summary metric), the knock-in and knock-out strategies consider each block a closed unit. They require no selection of a summary statistic. MI-based rankings deliver a score in $[0, 1]$, while VarGrad has no upper bound. However, the discretization associated with the mutual information calculation may influence the importance scores and, thus, the rankings by knock-in and knock-out. All strategies are applicable for multivariate target variables, as well. However, an MI-based comparison between outputs and pseudo-outputs is prone to suffer from the curse of dimensionality since higher-dimensional probability distributions are compared to each other. On the contrary, the vanishing gradient problem can influence VarGrad in deep ANN architectures. All approaches delivered accurate experimental results, but only knock-in and knock-out provided a consistent ranking of blocks with minor importance in dominant blocks, such as for the servo dataset.

Even though knock-in and knock-out rely on the same concept of assessing pseudo-outputs related to each block, their properties and interpretations differ. The knock-in strategy determines whether a block can deliver a reasonable target description independently from the remaining blocks. This interpretation of block importance evaluates the performance achieved if we reduce the model to solely one block at a time. In contrast, knock-out quantifies whether the contribution of a block can be compensated by any other block. If two blocks contain redundant information about the target, knock-in delivers high values for both blocks since each block individually has high explanatory power. In contrast, knock-out penalizes redundant blocks since each of them can be removed without loss of information. This property became evident in the BCW experiment, where B3 was dominant but shared overlapping information with B1: knock-in was the only approach that discovered the higher information content in B1 compared to the uninformative B2.

5 Conclusion

We have demonstrated three strategies to rank the importance of feature-blocks as post-processing in ANNs with block-wise input structure. The composite strategy, which is a direct generalization of feature-wise importance rankings, provided promising results in most cases, but selecting the correct summary statistic was crucial. Knock-in and knock-out strategies, implemented using an information-theoretic measure on the model outputs, delivered a trade-off

between the extremes of maximum and mean feature importance in the composite case. All methods uncovered the true block importance with high accuracy and delivered new insights into the ANN's behavior. Still, computing multiple proposed metrics is advantageous for making informative block ranking decisions.

Acknowledgment. The authors gratefully acknowledge the financial support from internal funding scheme at Norwegian University of Life Sciences (project number 1211130114), which financed the international stay at the University of British Columbia, and thereby fostered the completion of this work.

References

1. Adebayo, J., Gilmer, J., Goodfellow, I., Kim, B.: Local explanation methods for deep neural networks lack sensitivity to parameter values. arXiv (2018)
2. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity checks for saliency maps. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
3. Alnemer, L.M., et al.: Multiple sources classification of gene position on chromosomes using statistical significance of individual classification results. In: International Conference on Machine Learning and Applications and Workshops, vol. 1, pp. 7–12 (2011). https://doi.org/10.1109/ICMLA.2011.101
4. Amjad, R.A., Geiger, B.C.: Learning representations for neural network-based classification using the information bottleneck principle. IEEE Trans. Pattern Anal. Mach. Intell. **42**(9), 2225–2239 (2019). https://doi.org/10.1109/TPAMI.2019.2909031
5. Cao, B., He, L., Kong, X., Philip, S.Y., Hao, Z., Ragin, A.B.: Tensor-based multiview feature selection with applications to brain diseases. In: IEEE International Conference on Data Mining, pp. 40–49 (2014)
6. Cover, T., Thomas, J.: Elements of Information Theory. Wiley, Hoboken (2012)
7. Dagnely, P., Tourwé, T., Tsiporkova, E.: Annotating the performance of industrial assets via relevancy estimation of event logs. In: IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1261–1268 (2018). https://doi.org/10.1109/ICMLA.2018.00205
8. Ghorbani, A., Abid, A., Zou, J.: Interpretation of neural networks is fragile. In: AAAI Conference on Artificial Intelligence, vol. 33, pp. 3681–3688 (2019)
9. Hooker, S., Erhan, D., Kindermans, P.J., Kim, B.: A benchmark for interpretability methods in deep neural networks. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
10. Jenul, A., Schrunner, S., Liland, K.H., Indahl, U.G., Futsæther, C.M., Tomic, O.: Rent-repeated elastic net technique for feature selection. IEEE Access **9**, 152333–152346 (2021)
11. Jenul, A., Schrunner, S., Pilz, J., Tomic, O.: A User-Guided Bayesian Framework for Ensemble Feature Selection in Life Science Applications (UBayFS). arXiv (2021)
12. Quinlan, J.R.: Combining instance-based and model-based learning. In: International Conference on Machine Learning, pp. 236–243 (1993)

13. Street, W.N., Wolberg, W.H., Mangasarian, O.L.: Nuclear feature extraction for breast tumor diagnosis. In: Acharya, R.S., Goldgof, D.B. (eds.) Biomedical Image Processing and Biomedical Visualization, vol. 1905, pp. 861–870. SPIE (1993). https://doi.org/10.1117/12.148698
14. Wojtas, M., Chen, K.: Feature importance ranking for deep learning. Adv. Neural. Inf. Process. Syst. **33**, 5105–5114 (2020)
15. Yu, R., et al.: NISP: pruning networks borisusing neuron importance score propagation. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 9194–9203 (2018)
16. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. J. Roy. Stat. Soc. Ser. B (Stat. Methodol.) **68**(1), 49–67 (2006)

Paper III

Analytica Chimica Acta 1258 (2023) 341147

Contents lists available at ScienceDirect

Analytica Chimica Acta

journal homepage: www.elsevier.com/locate/aca

Non-linear shrinking of linear model errors

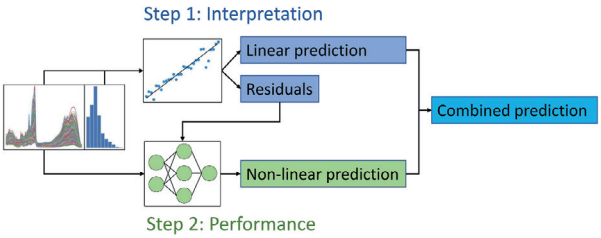
Runar Helin *, Ulf Indahl, Oliver Tomic, Kristian Hovde Liland

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

HIGHLIGHTS

- Residual shrinking is developed for both regression and classification problems.
- The proposed strategy can improve predictions while retaining interpretability.
- This contributes to explainable AI by shrinking the black box of ANNs.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Residual modelling
Hybrid model
Neural network
Interpretation
Deep learning
PLSR

ABSTRACT

Background: Artificial neural networks (ANNs) can be a powerful tool for spectroscopic data analysis. Their ability to detect and model complex relations in the data may lead to outstanding predictive capabilities, but the predictions themselves are difficult to interpret due to the lack of understanding of the black box ANN models. ANNs and linear methods can be combined by first fitting a linear model to the data followed by a non-linear fitting of the linear model residuals using an ANN. This paper explores the use of residual modelling in high-dimensional data using modern neural network architectures.

Results: By combining linear- and ANN modelling, we demonstrate that it is possible to achieve both good model performance while retaining interpretations from the linear part of the model. The proposed residual modelling approach is evaluated on four high-dimensional datasets, representing two regression and two classification problems. Additionally, a demonstration of possible interpretation techniques are included for all datasets. The study concludes that if the modelling problem contains sufficiently complex data (i.e., non-linearities), the residual modelling can in fact improve the performance of a linear model and achieve similar performance as pure ANN models while retaining valuable interpretations for a large proportion of the variance accounted for.

Significance and novelty: The paper presents a residual modelling scheme using modern neural network architectures. Furthermore, two novel extensions of residual modelling for classification tasks are proposed. The study is seen as a step towards explainable AI, with the aim of making data modelling using artificial neural networks more transparent.

* Corresponding author.
E-mail address: runar.helin@nmbu.no (R. Helin).

https://doi.org/10.1016/j.aca.2023.341147
Received 21 December 2022; Received in revised form 17 March 2023; Accepted 24 March 2023
Available online 4 April 2023

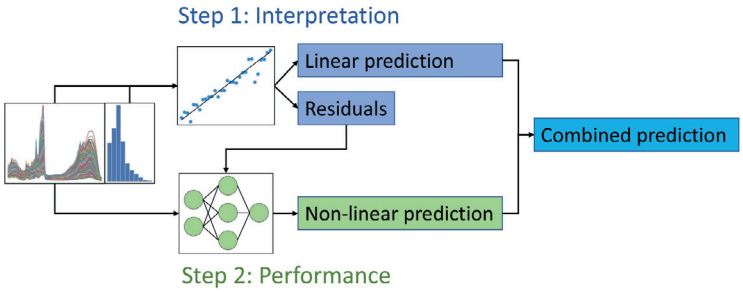


Fig. 1. Schematic illustration of the non-linear residual modelling. The input data is processed by a PLS model to generate interpretable linear prediction $\hat{\mathbf{y}}_{\text{PLS}}$. The linear prediction is subtracted from the true response to obtain the residuals $\mathbf{r}_{\text{PLS}}$ which are modelled by the ANN for improved performance of the hybrid model. The final prediction $\hat{\mathbf{y}}$ of the hybrid model is obtained as the sum of $\hat{\mathbf{y}}_{\text{PLS}}$ and $\hat{\mathbf{r}}_{\text{ANN}}$.

1. Introduction

Partial Least Squares (PLS) [1,2] is a popular method for linear regression model building and data analysis of high-dimensional spectroscopic and multicollinear data. The linear PLS models are considered to be relatively simple, and interpretable by considering the score- and loading vectors obtained from the model building process. In light of Beer Lambert's law, which states that the absorbance of a chemical species is proportional to the concentration of that species in a sample, linear models are usually assumed to be sufficient for modelling based on spectroscopic data modelling. Although often valid, effects caused by physical or chemical interferents, scattering effects or complex structured materials are sources that may violate Beer Lambert's law [3]. Recent studies have shown that non-linear models based on artificial neural networks (ANNs) can sometimes achieve better prediction performance than PLS models [4–6]. These findings lend credibility to the inclusion of neural networks in the standard toolbox for spectroscopic data analysis.

The rapid advances in deep learning technology have led to an increased interest in the study of neural networks and their potential for model building based on spectroscopic data. Despite this increased interest, research on problems related to image- and text analysis are still the dominant areas of application. By adapting techniques from established deep learning models, it has been demonstrated that convolutional neural networks (CNNs) are especially effective on 1D spectroscopic data [7–11]. Furthermore, as CNNs have achieved high prediction performance even without including explicit preprocessing of the raw spectra [5,12], there are indications that the effect of the preprocessing step is something that can actually be learned during the CNN learning process, given that relevant variation in the interfering signals is present in the spectra.

The ability to learn good feature representations automatically from the data is one of the strongest assets of ANN models. However, this comes at the cost of model transparency and makes the resulting neural network model weaker in terms of interpretations when compared to the linear models obtained by PLS and related methods. The troublesome understanding of ANN models is caused by the complex sequences of non-linear transformations for calculating the network outputs (predictions) from the input data.

In order to obtain useful interpretations along with the modelling there are two obvious alternatives: 1) One can discard ANN-based models and rely solely on interpretable models obtained by simpler (linear) strategies at the possible cost of some predictive performance. 2) One can focus on developing supplementary methodology to obtain more useful interpretations from ANN models, such as feature importances [13]. A third alternative is to consider a hybrid modelling approach that splits the model building process into a linear part (for interpretations) and a non-linear part (for enhanced model performance). In the literature, the latter approach has different names (both hybrid modelling and residual modelling have been used).

1.1. Comparison to the literature

Hybrid and residual modelling have been proposed in various forms. A popular approach to hybrid modelling is to use an ANN as a feature extractor [18]. This procedure can be used to extract non-linear features to later be modelled by other models, such as a PLS model [14]. More recently, large ANNs pre-trained on massive datasets of images are used to extract general features from images to be used as input for other models [19]. Another form of hybrid modelling is to use linear models to enrich the features fed to an ANN [15]. This is a kind of feature engineering approach to help the ANN model find useful relations in the data, and can be useful in situations with little available data. A third approach, and the one focused on in this study, is to use an ANN to model the residuals from a linear model [16]. The concept was first proposed to improve prediction performance of an ANN which suffered from poor generalisations [17]. The performance was improved by training the ANN to learn the difference between the linear model prediction and the response value and subsequently combine the linear and ANN predictions. Residual modelling has also been successfully applied to time series analysis based on the linear autoregressive integrated moving average (ARIMA) [20–22].

With increasing amounts of data collected and better deep learning technology, modern ANN models are often outperforming more traditional models. Thus, the residual modelling might not necessarily improve the prediction performance as was shown in the earlier studies. A more interesting avenue is therefore to use the residual modelling framework to better understand the problem and data. The present study explores this possibility of interpretation within the residual modelling framework. The framework is similar to Hussain et al. [17], but uses modern deep learning techniques such as rectified linear units (ReLU) activation functions and dropout layers [23] to obtain better performing ANN models. Different from previous studies, we apply modern ANN architectures to explore the effectiveness of non-linear modelling of the residuals from a linear model with a focus on maintaining model interpretability. The study focuses on high-dimensional spectroscopic data which has, to our knowledge, not previously been used with residual modelling. PLS and CNN models are used as examples of linear and deep learning models, respectively, because of their standing in the literature regarding interpretation possibilities and performance with spectral data [24,25]. Fig. 1 shows a basic illustration of the modelling scheme. Prediction of a new sample is obtained by adding the predicted residual term and the prediction from the linear model together.

In our study, the residual modelling is applied to four different high-dimensional datasets. We also present two novel approaches that extend the idea of doing residual modelling also for classification problems. The

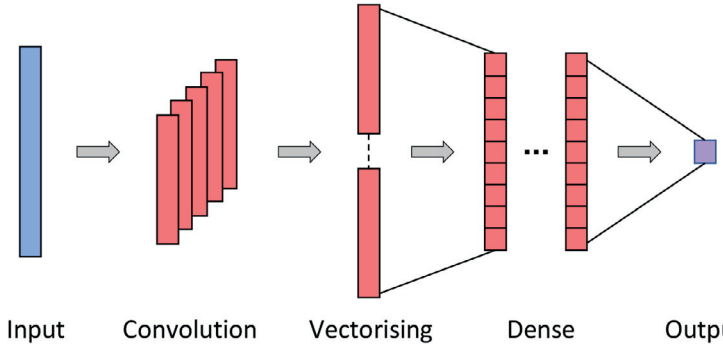


Fig. 2. Illustration of the base neural network architecture used for the 1D datasets.

predictive performance of the hybrid modelling scheme is compared to both pure PLS modelling and classical ANN modelling. Furthermore, possible diagnostic tools for model interpretations are presented and discussed for all four problems.

2. Methods and material

In the following, we give an overview of the models used in this study and a detailed description of the non-linear residual modelling approach.

2.1. Models

Predictive model building is a learning process for mapping data from some input space $\mathbb{R}^p$ to an output space $\mathbb{R}^c$. For the regression problems considered here, the output space is one-dimensional ($c = 1$) and for classification problems the number of dimensions c equals the number of groups.

In the following, let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be a data matrix with n measurements of p-dimensional samples. Assume that $\mathbf{X}$ is centred and possibly further preprocessed. Let $\mathbf{x}^T$ denote a row-vector from this matrix with corresponding response value(s) $\mathbf{y} \in \mathbb{R}^c$.

The prediction mapping of a linear model can be described by the equation

$$\widehat{\mathbf{y}} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{c\times p}$ and $\mathbf{b} \in \mathbb{R}^c$ are the parameters called *weights* and *biases*, respectively, in machine learning and *regression coefficients* and *intercepts* in statistics and chemometrics. In the case of a PLS model, equation (1) becomes $\widehat{\mathbf{y}} = \boldsymbol{\beta}\mathbf{x} + \bar{\mathbf{y}}$, where $\mathbf{W} = \boldsymbol{\beta}$ is a matrix with regression coefficients found through the PLS algorithm and $\bar{\mathbf{y}}$ are the mean response values over the training set. The PLS model projects the input data onto a low-dimensional latent space represented by score vectors $\mathbf{t}_a$. These score vectors are found sequentially as the linear combinations of the original features that maximise the covariance between an input matrix $\mathbf{X}$ and response $\mathbf{y}$. In the PLS model building, the score vectors are associated with corresponding loading vectors $\mathbf{p}_a$ which represent coordinates of the features in the compressed subspace. Together, the score- and loading vectors form rank one matrices which added together form the original data (maximum number of components) or an approximation: $\widehat{\mathbf{X}} = \sum_{a=1}^{A} \mathbf{t}_a \mathbf{p}_a^T$.

An ANN model is essentially a function that maps an input vector to some scalar or vector output through a sequence of non-linear transformations $f_1, \ldots, f_D$. The model is trained by adjusting the network parameters (weights) in a supervised fashion. For a network with layers $d = 1, \ldots, D$, each containing n_d nodes, the prediction of a sample $\mathbf{x}^T$ is obtained by the composed mapping

$$\widehat{\mathbf{y}} = (f_D \circ \cdots \circ f_1)(\mathbf{x}^T). \tag{2}$$

Each transformation is represented by a non-linear function (activation function) of the weighted sum of its vector input values, i.e., a non-linear vector-to-vector mapping associated with the nodes and weights between two *layers* of the network architecture. For example, the transformation associated with a fully connected layer d is defined as $f_d(\mathbf{z}^T) := \varphi(\mathbf{W}_d\mathbf{z} + \mathbf{b}_d)$, where $\varphi(\cdot)$ is the non-linear activation function operating element-wise on the argument vector. The weight matrix $\mathbf{W}_d$ can alternatively be replaced by a convolution matrix/operator to obtain a convolution layer. Each layer is associated with its own set of weights and biases that during the training process are adjusted iteratively by an error back-propagation gradient descent algorithm.

The most important goal of regression modelling is to identify model parameter values resulting in small prediction errors when applying the model. A popular strategy is to search for the model parameters minimising the mean squared error (MSE) of the training data: $\| \mathbf{y} - \widehat{\mathbf{y}} \|^2/n$, where n is the number of samples and $\widehat{\mathbf{y}}$ are the associated model predictions.

In PLS regression, the objective implemented by the algorithm includes decomposition and dimension reduction of the data matrix $\mathbf{X}$ followed by linear least squares modelling with the reduced data. For a neural network the objective is chosen as a loss function (cost function) to be minimised by a gradient descent search. In classification problems, the objective is to minimise the number of misclassified samples. For a classification problem including c different classes (groups), and an n-dimensional vector representing the class labelling, the latter is usually replaced by a (one-hot encoded) $\mathbf{Y} \in \mathbb{R}^{(n\times c)}$ dummy matrix.

In the PLS modelling approach to classification the dummy matrix is taken as the response data to obtain the model predictions $\widehat{\mathbf{Y}}$. Thereafter, a classification model can be obtained by linear discriminant analysis (LDA) [26] of the fitted values $\widehat{\mathbf{Y}}$.

The dummy matrix $\mathbf{Y}$ can also be taken as the responses for training an ANN model with c output nodes including softmax transformations of the model output to produce class membership probabilities. A categorical cross-entropy loss function, defined by

$$L(\mathbf{Y}, \widehat{\mathbf{Y}}) = -\sum_{i=1}^{n}\sum_{j=1}^{c} Y_{ij} \ln \widehat{Y}_{ij} \tag{3}$$

is most often used for training the classification ANN, where $\widehat{Y}_{ij}$ is the predicted probability of sample i to belong to class j.

2.2. Neural network architecture and training

Choosing an efficient neural network architecture is challenging

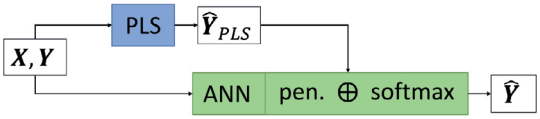


Fig. 3. Illustration of the CE-shrinking approach. The penalised contribution of the ANN prediction and the PLS prediction are added together (element-wise) before the softmax activation in the network. Note that the PLS prediction is unchanged during the ANN training phase.

given the almost unlimited number of possible network configurations. One of the most important aspects is to find the correct model complexity/capacity. The network must be sufficiently complex to model the data. However, inclusion of too many weights may lead to model overfitting. Due to limited research on network architectures for 1D spectroscopic data, there are no clear recommendations for the choice of model architectures, often resulting in a large amount of time spent on model tuning.

In the present paper, our choice(s) of architecture(s) are based on prior experience reported in the literature [12,27]. An illustration is shown in Fig. 2.

This architecture includes one convolution layer followed by a stack of fully connected (dense) layers. A non-linear activation function is used element-wise in the mappings between each layer, and the dropout principle [23] is used for the mappings into the final layer. The number and size of the convolution filters, the number of fully connected layers, the number of nodes in each hidden layer, the type of activation function and the dropout rate are all to be considered as hyperparameters.

The hyperparameters control the complexity of the model and are tuned to each specific dataset in the present work using a Bayesian optimisation [28]. A Python algorithm utilising the Keras library of TensorFlow [29] implements the training process. Specifications of the ranges used in the hyperparameter search are provided in the Supplementary materials. The computations were executed on an NVIDIA Quadro RTX 8000 graphics processing unit (GPU). The PLS and LDA modelling were done using Python and the scikit-learn package version 1.1.1 [30].

The ANNs used for model predictions and non-linear residual modelling on 1D data use the base architecture described above. Among the datasets there is one set containing 2D images, for which the ANN used is a standard 2D CNN suitable for image recognition. This 2D architecture consists of blocks of convolution layers and batch normalisation layers with one fully connected layer and regularising dropout before the output layer. Dimension reduction is obtained by strided convolutions between some network layers. For more details on the architecture, see the Supplementary materials.

The ANN training is based on model updating during multiple epochs of error back-propagation. The number of epochs (cycles through the training data) is essential for tuning the networks as it affects the amount of over/under fitting. Determination of the appropriate number of epochs for training is usually done by monitoring the loss (prediction error) on an independent validation set during the training phase.

2.3. Non-linear residual modelling

The concept behind residual modelling is to improve the prediction performance of a linear model by explicitly modelling the prediction errors (residuals) from that linear model using a neural network. This idea is most straight-forward for regression problems, but the same concept can be applied on classification problems (described below). The residual modelling procedure is illustrated in Fig. 1. First, the input data ($\mathbf{X}$, $\mathbf{y}$) are used to train a PLS model which generates the linear prediction $\hat{\mathbf{y}}_{\text{PLS}}$. Second, an ANN is trained using the residuals $\mathbf{r}_{\text{PLS}} = \mathbf{y} - \hat{\mathbf{y}}_{\text{PLS}}$ as target for the network. Note that the network takes the same data $\mathbf{X}$ as input. Denoting the prediction of the residuals $\hat{\mathbf{r}}_{\text{ANN}}$, the final prediction of the non-linear residual shrinking problem is the sum $\hat{\mathbf{y}} = \hat{\mathbf{y}}_{\text{PLS}} + \hat{\mathbf{r}}_{\text{ANN}}$.

2.3.1. Classification problems

It is not as obvious how to transform the residual modelling from minimising continuous errors to correcting categorical misclassifications. We explore and compare two different approaches called MSE-shrinking and CE-shrinking (described below) representing different ways to utilise the ANN to correct the linear model outputs.

In the MSE-shrinking approach, the ANN is trained to learn the matrix of residuals from the multi-class PLS prediction $\mathbf{R} = \mathbf{Y} - \hat{\mathbf{Y}}_{\text{PLS}}$. This approach is similar to regression problems where the network tries to learn the difference between the one-hot encoded matrix and the PLS prediction. The final prediction is obtained by an LDA model trained on the sum $\hat{\mathbf{Y}}_{\text{PLS}} + \hat{\mathbf{R}}_{\text{ANN}}$. The loss function of this network is MSE, hence the name MSE-shrinking.

In the MSE-shrinking approach, the ANN is essentially doing a regression and is unaware of the final goal of classification. In order to obtain an objective more relevant for classification problems, we also considered the CE-shrinking alternative where the network is using $\hat{\mathbf{Y}}_{\text{PLS}}$ as additional inputs. The purpose of this approach is to try to learn the residuals implicitly by forcing the network to improve on the provided PLS predictions. This is achieved by adding the PLS prediction $\hat{\mathbf{Y}}_{\text{PLS}}$ to the output of the last layer prior to the softmax activation as illustrated in Fig. 3. Since the loss function of this network is the commonly used categorical cross-entropy, we refer to this alternative residual modelling approach as CE-shrinking.

Upon testing, it turned out that the networks required some constraints to avoid ignoring the provided PLS predictions. Without constraints, the network models had a tendency to make the outputs of the ANN block shown in Fig. 3 extremely large. In effect this essentially made the provided PLS predictions irrelevant.

To overcome this problem, we included an L2-regularisation to the ANN contribution by adding $\lambda \parallel \mathbf{Z} \parallel$ to the loss function, where $\mathbf{Z}$ is the pre-softmax output of the ANN block. By this, the values are forced to be of a magnitude similar to the PLS predictions. Through visual inspection, it was found that $\lambda = 1$ constrained the network modelling sufficiently but not too much. Since the $\hat{\mathbf{y}}_{\text{PLS}}$ will have values close to the range 0–1, it is reasonable to assume that $\lambda = 1$ is a good choice regardless of the dataset. An alternative to visual inspection is to compare the Frobenius norms of $\hat{\mathbf{Y}}_{\text{PLS}}$ and $\mathbf{Z}$ to determine the appropriate value of λ. The parameter λ acts as an additional hyper-parameter that control the trade-off between interpretation and performance. With insufficient regularisation (λ chosen to be too small), the PLS model interpretations of our hybrid modelling approach may be invalid.

2.3.2. Model selection and validation

The non-linear residual modelling approach was tested on four high-dimensional benchmark datasets described in the next section. During the analysis, all datasets, except the MNIST data were divided into training (75%) and test (25%) sets in a stratified manner where applicable. The test sets were held aside for the final model evaluations. The MNIST dataset is an exception since it comes with predefined sets of training- and test data.

The regression problems were evaluated using the R^2 metric and the classification problems by prediction accuracy (i.e., percentage of correctly classified samples). During training of the ANNs, the MSE was minimised.

2.3.2.1. Model selection. Both the PLS and ANN models require tuning of model complexity. This complexity was tuned for each individual dataset. For the PLS model, the optimal number of components was found using a 5-fold cross validation on the training partition of the data. The optimal network architecture for each problem was found using the Bayesian optimisation framework described earlier, with 1/3 of the

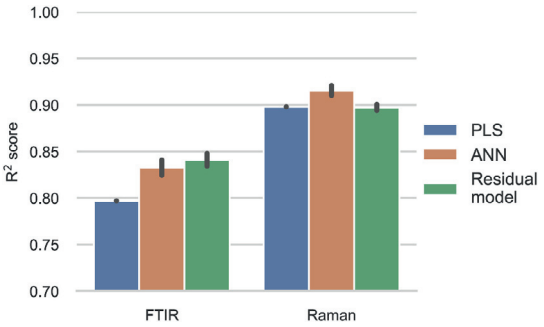


Fig. 4. Regression problems. ANN results are the means of 10 trials with standard deviation as the error bar. The y-axis is truncated.

training data as a validation set. Using a single partition of the training set reduces computational cost drastically compared to a full cross validation procedure. We have taken the point of view that a model predicting well on the validation data is also expected to perform well on the test set.

2.3.2.2. Model evaluation. During evaluation of the models, the PLS model was trained on the whole training set (75% of the total amount of data) and evaluated on the test set. The ANN model was trained on 2/3 of the training data where the final 1/3 was used as a validation set to determine how many epochs to train the ANN before convergence. The trained network was then used to evaluate the test set. Another practice included in our experiments was to do the test set evaluation based on the average of 10 neural network models using different random weight initialisations in order to obtain more robust estimates of the expected predictive performance.

Fitting the models in the non-linear residual modelling, had to be done with care. The PLS model training residuals will typically be smaller in magnitude than the validation residuals as the same samples are used for training and prediction. This means the ANN would be trained on unrealistic residual magnitudes if applied directly to the training residuals. A 5-fold cross-validation was therefore used to obtain more realistic estimates of the residuals from the PLS model before training the ANN using the cross-validated residuals as responses. After training the residual ANN on the cross-validated residuals, again using 1/3 of the data to determine convergence of the network, the test set predictions were made by first retraining the PLS model on the whole training dataset to obtain the linear test set predictions $\hat{\mathbf{y}}_{\mathrm{PLS-test}}$. Thereafter, the ANN model trained on the cross-validated residuals was used to obtain the non-linear predictions $\hat{\mathbf{r}}_{\mathrm{ANN-test}}$ of the test set.

2.4. Datasets

2.4.1. FTIR

The first dataset is for regression and contains Fourier-Transform Infrared (FTIR) spectra obtained from enzymatic protein hydrolysis processes of different rest raw materials from food production [31]. The raw materials come from fish and poultry, such as fish heads and chicken mechanical deboning residue, which were hydrolysed by different kinds of enzymes. As the response, the average molecular weight (AMW) of proteins was used, which works as a proxy for the degree of hydrolysis. In total, the dataset contains 885 spectra with a varying number of replicates. In the subsequent analysis, the spectra with wavenumbers between 1800 cm^{-1} and 700 cm^{-1} are used, resulting in 571 features for each spectrum. Prior to the modelling, the spectra were preprocessed using Savitzky-Golay smoothed second derivatives with filter width of 11 points and polynomial smoothing of 3rd degree followed by extended multiplicative signal correction (EMSC) with 2nd degree polynomial baseline correction. As an extra preprocessing step for the neural networks (both ANN modelling and residual shrinking), the data was standardised column-wise (autoscaling).

2.4.2. Raman

The second dataset is for regression and contains Raman spectra from samples of milk [32]. The dataset contains 2682 spectra with varying numbers of replicates. The wavenumber range is between 3100 cm^{-1} to 120 cm^{-1}, resulting in 2979 features for each spectrum. Each spectrum was preprocessed using EMSC with 6th order polynomial baseline correction. No further preprocessing was done prior to the neural networks for this dataset.

2.4.3. NIR

The third dataset is a classification problem and contains Near Infrared (NIR) spectra from a remote sensing hyperspectral image over an area covering 16 different vegetation and soil types in Salinas Valley, California [33]. The spectra cover the wavelength range 400 nm–2500 nm and have 204 features each. In our experiment, a random subset of 200 samples per class was used. Prior to the analysis, the hyperspectral bands corresponding to water absorption were removed. As a preprocessing step, the Standard Normal Variate (SNV) was performed. No further preprocessing was done prior to the neural networks.

2.4.4. MNIST

The final dataset is the Modified National Institute of Standards and Technology database (MNIST) dataset consisting of 28 × 28 pixel greyscale images of handwritten digits [34] used for classification. For the PLS model, we convert the samples to 1D by treating each pixel as a feature, resulting in 784 features in total. This dataset contains 70 000 samples (60 000 for training and 10 000 for testing). The classes are relatively balanced with the smallest and largest classes containing 9.02% and 11.25% of the total data respectively. Each feature was normalised to have values between 0 and 1. No further preprocessing was made for any of the models. A 2D CNN is better suited to capture features in the images since it also makes use of the spatial information in the data. Therefore, a 2D CNN was applied with the original input data shape of 28 × 28 pixels for both ANN modelling and residual shrinking.

3. Results

The prediction performance of each model is summarised in Figs. 4 and 5. Starting with the regression problems and the FTIR dataset, the best performance was achieved by the non-linear residual model with an R^2 score of 0.841. The ANN was slightly (but not significantly) worse with an R^2 score of 0.833. Both alternatives clearly outperformed the PLS model which got an R^2 score of 0.797. With the Raman dataset, the pure ANN performed slightly better than the PLS with R^2 scores of 0.915 and 0.898 respectively. In contrast to the FTIR dataset, the non-linear residual modelling did not improve on the linear prediction. The PLS models for the FTIR and Raman datasets included 28 and 16 components, respectively.

For the classification datasets we used prediction accuracy (proportion correctly classified) as performance measure since the datasets were relatively balanced. For the NIR data, the PLS model achieved an accuracy of 0.958. The ANN had the lowest accuracy of 0.955 while the MSE- and CE-shrinking methods had accuracies of 0.958 and 0.963, respectively. Taking the uncertainties into account, one cannot claim that any of the neural network based methods outperforms the PLS on this dataset, meaning that the PLS model was sufficient for this task. On the MNIST dataset, the PLS model got an accuracy of 0.877 while the pure ANN got an accuracy of 0.992. The MSE-shrinking and CE-shrinking techniques got accuracies of 0.994 and 0.986, respectively, both significant improvements over the underlying PLS model. The PLS

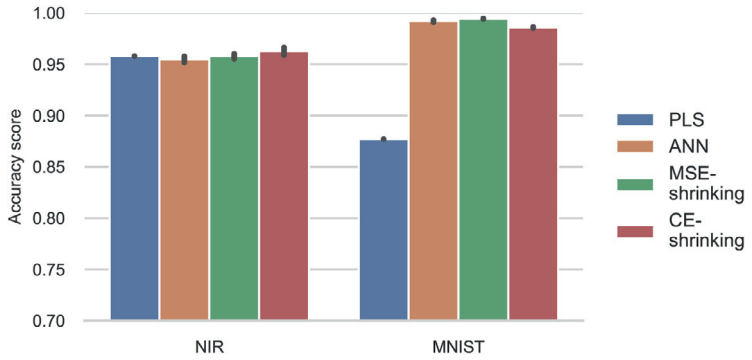


Fig. 5. Classification problems. ANN results are the means of 10 trials with standard deviation as the error bar. The y-axis is truncated.

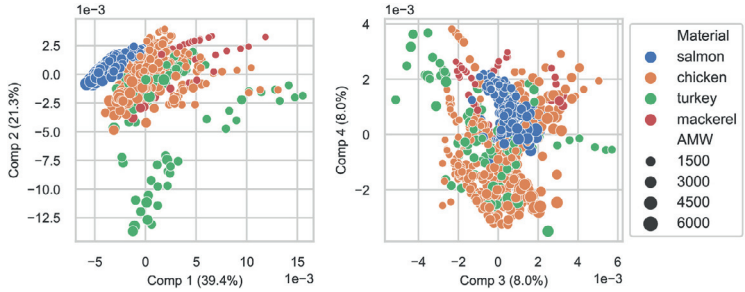


Fig. 6. Score plots from the PLS model of the FTIR dataset. Explained feature variance is shown in parentheses, raw material are encoded as colours and response values as circle sizes. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

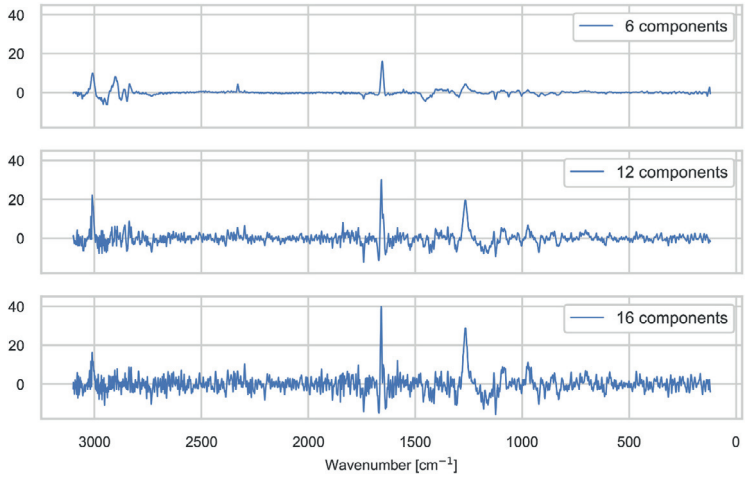


Fig. 7. Regression coefficients from PLS models trained in the Raman milk dataset using 6, 12 and 16 components.

models for the NIR and MNIST datasets used 20 and 27 components, respectively. The relatively high number of components needed must be seen in light of the large number of classes (16 for the NIR data) and the high intra-class variation (various ways of writing the same digits).

3.1. Interpretation

The main motivation for applying non-linear residual modelling was to utilise the power of neural networks to model non-linear relationships, while retaining the interpretability. Since the residual modelling

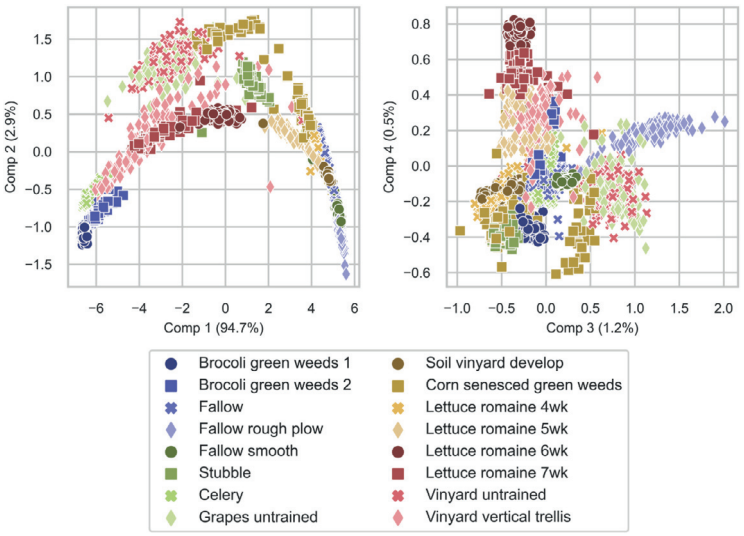


Fig. 8. Score plots from the PLS model of the remote sensing NIR dataset. Explained feature variance is shown in parentheses and symbol usage indicates different classes.

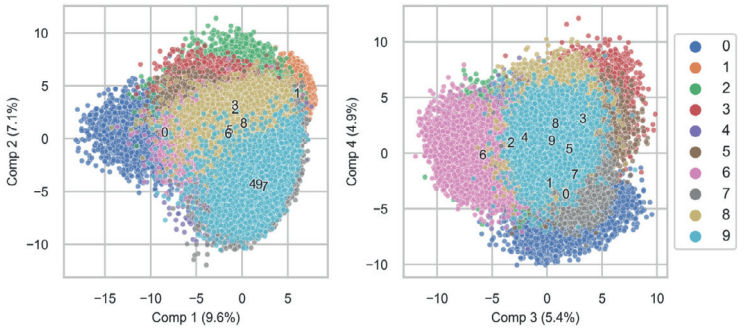


Fig. 9. Score plots from the PLS model trained on the MNIST dataset. Explained feature variance is shown in parentheses, and symbol colours indicate digits. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

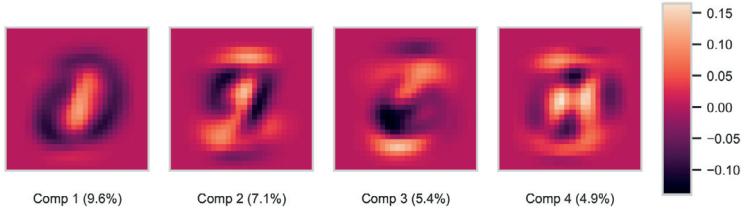


Fig. 10. Loading weights of the first four PLS components from the MNIST dataset, reshaped back to original image size (28 × 28).

does not affect the linear prediction part, all interpretations of the PLS model are still to be considered as valid. In the following we explore a selection of possible graphical diagnostic tools and their purpose. Many of these diagnostic tools are not available in pure ANN modelling and highlights the usefulness of the residual modelling approach.

Fig. 6 shows score plots of the first four PLS components from the model trained on the FTIR dataset. This kind of plot highlights patterns in the latent space, shown by similarities or clusters of samples in the scatter plots. Based on the observed groupings, it is sometimes possible to explain which samples share chemical or physical properties in the corresponding components. For instance, by colouring the samples in this dataset by their product group (chicken, turkey, mackerel and

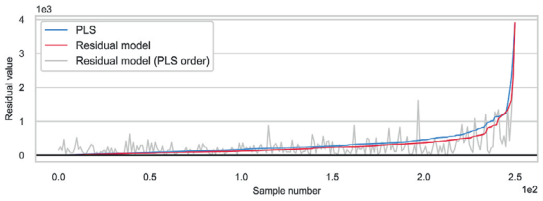


Fig. 11. Line plot of the test set residuals (absolute values) from the FTIR hydrolysates dataset for the pure PLS model and after the residual modelling. The blue and red lines show the PLS model residuals and the new residuals after the residual shrinking respectively in increasing order from left to right. The grey line shows the new residuals after the residual shrinking in the same order as the sorted PLS residuals. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

salmon), it is confirmed that the first two components distinguishes salmon from other samples. The plot also emphasises a group of turkey samples with larger AMW values. This demonstrates that from such score plots one can detect possibly unknown groupings or confirm already known or hypothesised groupings in the data. Furthermore, such plots can also be useful for detection outliers.

Fig. 7 shows line plots of the regression coefficients obtained from three different PLS models including 6, 12 and 16 PLS components, respectively, all trained on the Raman milk dataset. Such plots are useful for detecting important regions in the feature space for predicting the response, indicated by positive or negative peaks and contrasts. The model including 6 PLS components shows a peak in the region around the 1700 cm^{-1} shift. This peak is typical in systems with rich lipid content such as these samples [32] and can be an indication of varying lipid contents along with the iodine value variation. Other areas of interest for similar substances include 3000 cm^{-1} and 1300 cm^{-1}, where indeed peaks are also observed. However, more than 6 PLS components are needed before these become visually apparent. Another observation is that the regression coefficients get more noisy with an increased number of components and is an indication of model over fitting.

For the NIR remote sensing dataset, the score plots shown in Fig. 8 reveal important insights. The first two components (left figure) displays a curvy structure which indicates that there may be some effect in the spectra picked up by the model that should have been corrected for in the preprocessing. A hypothesis for explaining this phenomenon is that it has something to do with the pre-treatment of removing the water-absorbance part of the spectra prior to the analysis. In both plots we get a visual confirmation of the similarity of samples within the same category as well as indication of outlying samples, e.g., the Vinyard vertical trellis observation at around coordinates (2,-0.5) in the leftmost plot.

For the MNIST dataset, the associated score plots in Fig. 9 show how similarities between the different classes can be revealed. The score plot of the first two components, shows that digits with similar appearance such as 4, 9 and 7 tend to be grouped closely together. Furthermore, samples of the digit 1 lie far away from samples of digit 0, indicating that the first component distinguishes curves from straight lines. Another interesting observation is that the amount of explained variance is fairly low for the earlier components compared to the models for the other datasets, meaning that PLS modelling requires more components to effectively capture all variance of the MNIST features. Since the MNIST dataset consists of 2D sample images, it is also possible to visualise the PLS loading weights as images in the original 28 × 28 pixel image space as shown in Fig. 10. The loading weights highlight regions in the input space with large impact on the resulting PLS model(s) and associated

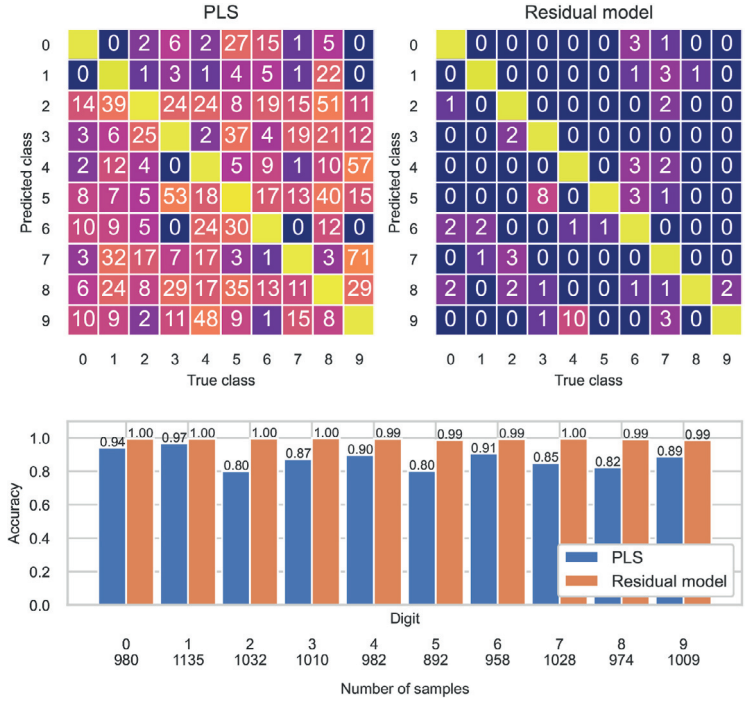


Fig. 12. Confusion matrices of test set predictions on the MNIST dataset. Left: PLS model, right: PLS + non-linear residual modelling. The numbers in the diagonals are removed for clarity and the information is summarised in the bar plot at the bottom. The bar plot shows the classification accuracy per class for each model.

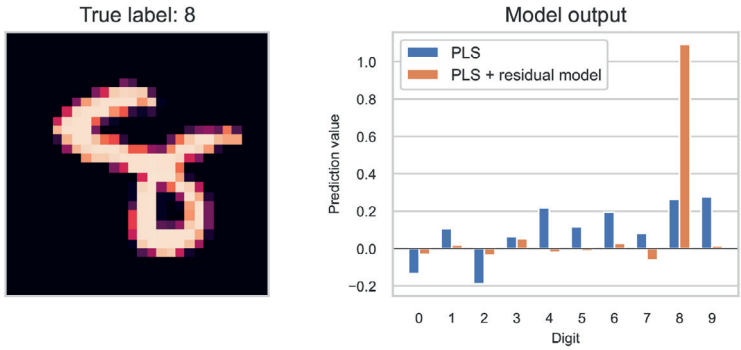


Fig. 13. Output from PLS model and after the MSE-shrinking approach for a sample image. True label: 8, PLS prediction: 9, prediction after residual shrinking: 8.

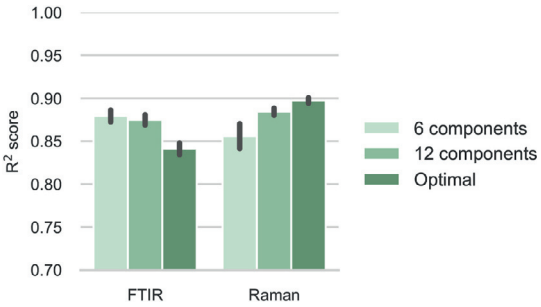


Fig. 14. R^2 score of the non-linear residual modelling using different numbers of PLS components. The error bars indicate the standard deviation of 10 different ANN weight initialisations.

predictions. The first PLS loading seems to be focused on capturing oval shapes like zeros consistent with the earlier interpretation from the score plot. The other loadings also resemble structures of familiar handwritten numbers.

The residual modelling approach offers additional options regarding model interpretation by studying how the final residuals are changed compared to the PLS residuals. For regression problems, the changes can be visualised as line plots as in Fig. 11. In this figure, the absolute values of the residuals of the test set predictions are shown. The blue line in the figure corresponds to the PLS model residuals plotted in increasing order from left to right. The corresponding new residuals after the residual shrinking are shown by the grey line. The red line corresponds to these new residuals after the residual shrinking in increasing order. It can be observed that the PLS residuals with large absolute values tend to get reduced by the ANN, while some samples get a higher residual after performing the residual modelling. The plot reveals the gain of the residual modelling, and may be useful for identifying possible outliers, e.g., samples with notably larger residuals compared to the overall residual distribution. The indicated outlying samples can then be inspected with respect to eventual irregularities in the data collection procedure.

In classification problems, a simple diagnostic tool to study the effect of the residual modelling are confusion matrices as shown in Fig. 12. A confusion matrix is used to evaluate a model performance by giving a summary of the counts of true versus predicted classes. By convention, the confusion matrix rows indicate the true class and the columns indicate the predicted class. For a useful classification model, the diagonal entries, showing the number of correctly classified samples, should contain the largest numbers. Here, these numbers are summarised as a bar plot for good overview. The confusion matrices reveal classes that are more challenging than other. For instance, the PLS model predicts the digit 7 to be a 9 wrongly 71 times. However, this mistake, along with most others, are corrected by the ANN model. Additionally, one can look at individual samples which the PLS model predicted wrongly and corrected by the residual shrinking, with the aim to pinpoint where the linear model works poorly. An example image is shown in Fig. 13 where the PLS model predicted the digit 9 instead of the correct digit 8. The barplot shows that the PLS model was not very confident in its decision among the alternative candidates (8, 4 and 6). Interestingly, the model

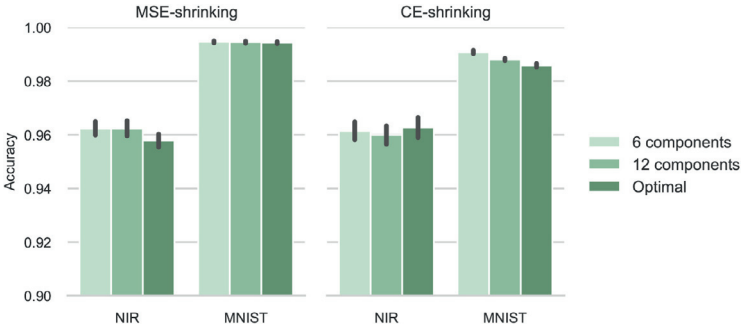


Fig. 15. Accuracy score of the non-linear residual modelling for classification using different numbers of PLS components. The error bars indicate the standard deviation of 10 different ANN weight initialisations.

obtained after the residual estimation is extremely confident in its prediction of the class to be an 8.

4. Discussion

The main benefit of the presented non-linear residual model scheme is that model interpretations are partly available, i.e., from the linear part of the modelling. In cases where the linear part of a hybrid model dominates the contribution to prediction it also provides better insights than the pure ANN modelling alternative. The examples shown in the results section indicate that non-linear ANN-modelling of the linear PLS residuals yields models that are often close to the pure ANN models in performance. The utility of the non-linear residual modelling becomes most evident when the problem is sufficiently complex and may serve as a useful alternative to a black-box ANN.

Our results also indicate that ANN based models are not always the superior choice regarding good predictions. For both the Raman- and NIR datasets analysed in this study, the difference in predictive performance between the ANN- and PLS models were insignificant. For such simple problems, it is therefore not surprising that the idea of hybrid modelling adds little or nothing with regard to improved predictive performance.

It is noteworthy that both the MSE-shrinkng and CE-shrinking approaches for classification performed well despite differences in the corresponding optimisation problems (i.e., different loss functions). No conclusion can be drawn whether one of the two approaches is superior based on the examples considered here. One major difference between the two alternatives is that the CE-shrinking requires tuning of an additional hyperparameter to set the balance between how much the PLS- and the ANN model contribute to the hybrid model predictions.

A fair question to ask is whether the proposed hybrid non-linear residual modelling is necessary or if one alternatively could train a PLS model to support interpretations and a separate ANN model to handle the predictions. However, this alternative provides no direct connection between the separate PLS- and ANN model behaviours. With the hybrid residual modelling scheme it will always be possible to inspect a prediction to assess its composition with regard to the linear and non-linear contributions. In effect, the proposed hybrid modelling can be thought of as shrinking the size of the "black box" associated with the ANN model, where the shrinkage depends on the dominance of the linear model part.

Supported by prior knowledge regarding the data, subject to the model building, it is possible to judge the validity of a model by considering the content of the different model visualisation techniques. The diagnostic tools for interpretation presented can roughly be divided into two categories. On the one hand, the score- and loading plots provide insight into the subspace spanned by the PLS components. The other type of visualisation is provided by filters and feature maps of the neural network and governs a different feature space than the PLS model. The choice of visualisation from the ANN is dependent on the network architecture, and currently convolutional layers seem to offer the best options. Currently, it is easier to relate to visualisation of images as shown with the MNIST dataset. But, 1D signals can be treated in a similar fashion to detect regions of higher activation [7].

The residual modelling is not the only way of modelling non-linearities with a PLS model. Alternatively, a non-linear extensions of PLS modelling such as RBF-PLS [35] can be considered. This model might be a good alternative but lack the same flexibility in feature representations the ANN provides. Interpretability through the use of kernel functions might also be challenging.

Since the MNIST dataset consists of 2D images forming a three-dimensional tensor representation of input data, $\underline{\mathbf{X}}$, an alternative to vectorising (unfolding) the images to be features for ordinary PLS modelling is the application of a multilinear PLS version like N-PLS or N-CPLS [36,37] directly with the tensor representation. These methods also provide loading weights and loadings along each of the image dimensions which can be beneficial for understanding the resulting model. However, when comparing these modelling alternatives for the MNIST data, performance was near identical to ordinary PLS.

In our examples, non-linear residual modelling was applied on PLS models using the optimal number of components decided by cross-validation. It can be argued that using fewer components might be beneficial for the subsequent residual modelling approach. With fewer PLS components, the neural network essentially gets more freedom since the PLS model does not explain as much variance. Furthermore, since interpretation is typically done using the first few components, much of the insights are retained. Using fewer PLS components might also reduce the risk of letting the PLS model attempt to model non-linearities more suited for the ANN. To test the effect of using different numbers of components, all examples were repeated using 6 and 12 PLS components respectively, and keeping everything else unchanged, while repeating the ANNs on the new and larger residuals. A presentation of the results to be compared are shown in Figs. 14 and 15. In general, the use of fewer PLS components resulted in better performance of the resulting hybrid model. An exception is the Raman dataset where the use of fewer PLS components gave significantly worse performance. A possible explanation for this is that the ANN training process was unable to find any relevant information in the residuals. Hence, the ANN was not able to successfully model the larger residuals from a reduced PLS model (fewer components), resulting in an overall worse performance compared to an optimal PLS model. The failure of the residual modelling is shown by the higher variance in the associated error bar for the 6 component model alternative.

5. Conclusion

We have demonstrated a hybrid modelling framework where an artificial neural network (ANN) is used to predict residuals from a linear model. The concept is presented for regression problems and later extended for classification tasks. In the residual modelling scheme, the data modelling is split into a linear and non-linear part, allowing the majority of the data modelling to be done by a linear interpretable model while the ANN is used to boost the prediction performance. It is shown that with the proposed framework, it is possible to achieve almost the same predictive performance as a pure ANN modelling but with the added benefit that a larger proportion of the model can be interpreted, thereby shrinking the black box of the ANN.

CRediT authorship contribution statement

Runar Helin: Methodology, Software, Writing – original draft. **Ulf Indahl:** Conceptualization, Methodology, Supervision, Writing – review & editing. **Oliver Tomic:** Methodology, Supervision, Writing – review & editing. **Kristian Hovde Liland:** Conceptualization, Methodology, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.aca.2023.341147.

References

[1] Svante Wold, Harold Martens, Herman Wold, The multivariate calibration problem in chemistry solved by the PLS method, in: Matrix Pencils, Springer, 1983, pp. 286–293.

[2] S. Wold, A. Ruhe, H. Wold, W.J. Dunn, The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses, SIAM J. Sci. Stat. Comput. 5 (1984) 735–743.

[3] M. Mamouei, K. Budidha, N. Baishya, M. Qassem, P.A. Kyriacou, An empirical investigation of deviations from the beer–lambert law in optical estimation of lactate, Sci. Rep. 11 (1) (2021), 13734–13734.

[4] A. Kasper, Einarson, Andreas Baum, Terkel B. Olsen, Jan Larsen, Ibrahim Armagan, Paloma A. Santacoloma, Line K.H. Clemmensen, Predicting pectin performance strength using near-infrared spectroscopic data: a comparative evaluation of 1-d convolutional neural network, partial least squares, and ridge regression modeling, J. Chemometr. 36 (2) (2022).

[5] Runar Helin, Ulf Geir Indahl, Oliver Tomic, Kristian Hovde Liland, On the possible benefits of deep learning for spectral preprocessing, J. Chemometr. 36 (2) (2022).

[6] Puneet Mishra, Dário Passos, Multi-output 1-dimensional convolutional neural networks for simultaneous prediction of different traits of fruit based on near-infrared spectroscopy, Postharvest Biol. Technol. 183 (2022).

[7] Jacopo Acquarelli, Twan van Laarhoven, Gerretzen Jan, Thanh N. Tran, M. Lutgarde, C. Buydens, Elena Marchiori, Convolutional neural networks for vibrational spectroscopic data analysis, Anal. Chim. Acta 954 (2017) 22–31.

[8] Uladzislau Blazhko, Volha Shapaval, Vassili Kovalev, Achim Kohler, Comparison of augmentation and pre-processing for deep learning and chemometric classification of infrared spectra, Chemometr. Intell. Lab. Syst. 215 (2021).

[9] Xiaolei Zhang, Tao Lin, Jinfan Xu, Xuan Luo, Yibin Ying, Deepspectra: an end-to-end deep learning approach for quantitative spectral analysis, Anal. Chim. Acta 1058 (2019) 48–57.

[10] Salim Malek, Farid Melgani, Yakoub Bazi, One-dimensional convolutional neural networks for spectroscopic signal regression, J. Chemometr. 32 (2018), e2977.

[11] Jialin Dong, Mingjian Hong, Yi Xu, Xiangquan Zheng, A practical convolutional neural network model for discriminating Raman spectra of human and animal blood, J. Chemometr. 33 (2019), e3184.

[12] Chenhao Cui, Tom Fearn, Modern practical convolutional neural networks for multivariate regression: applications to NIR calibration, Chemometr. Intell. Lab. Syst. 182 (2018) 9–20.

[13] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, Been Kim, A benchmark for interpretability methods in deep neural networks, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019.

[14] Greger Andersson, Peter Kaufmann, Lars Renberg, Non-linear modelling with a coupled neural network - pls regression system, J. Chemometr. 10 (5–6) (1996) 605–614.

[15] Mehdi Khashei, Ali Zeinal Hamadani, Mehdi Bijari, A novel hybrid classification model of artificial neural networks and multiple linear regression models, Expert Syst. Appl. 39 (3) (2012) 2606–2620.

[16] Peigen Yu, Mei Yin Low, Weibiao Zhou, Development of a partial least squares-artificial neural network (pls-ann) hybrid model for the prediction of consumer liking scores of ready-to-drink green tea beverages, Food Res. Int. 103 (2018) 68–75.

[17] M.A. Hussain, M. Shafiur Rahman, C.W. Ng, Prediction of pores formation (porosity) in foods during drying: generic models by the use of hybrid neural network, J. Food Eng. 51 (3) (2002) 239–248.

[18] Suresh Dara, Priyanka Tumma, Feature extraction by using deep learning: a survey, in: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018, pp. 1795–1801.

[19] Mohammed Brahimi, Kamel Boukhalfa, Abdelouahab Moussaoui, Deep learning for tomato diseases: classification and symptoms visualization, Appl. Artif. Intell. 31 (4) (2017) 299–315.

[20] G. Peter, Zhang. Time series forecasting using a hybrid arima and neural network model, Neurocomputing 50 (2003) 159–175.

[21] Phatchakorn Areekul, Tomonobu Senjyu, Hirofumi Toyama, Atsushi Yona, A hybrid arima and neural network model for short-term price forecasting in deregulated market, IEEE Trans. Power Syst. 25 (1) (2010) 524–530.

[22] Guoying Wang, Lili Zhuang, Lufeng Mo, Xiaomei Yi, Peng Wu, Xiaoping Wu, Bag: a linear-nonlinear hybrid time series prediction model for soil moisture, Agriculture 13 (2) (2023) 379.

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (56) (2014) 1929–1958.

[24] David M. Haaland, Edward V. Thomas, Partial least-squares methods for spectral analyses. 1. relation to other quantitative calibration methods and the extraction of qualitative information, Anal. Chem. 60 (11) (1988) 1193–1202.

[25] Puneet Mishra, Dário Passos, Federico Marini, Junli Xu, Jose M. Amigo, Aoife A. Gowen, Jeroen J. Jansen, Alessandra Biancolillo, Jean Michel Roger, Douglas N. Rutledge, Alison Nordon, Deep learning for near-infrared spectral data modelling: hypes and benefits, TrAC, Trends Anal. Chem. 157 (2022).

[26] Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning. Springer Series in Statistics, Springer, New York, 2009.

[27] Dário Passos, Puneet Mishra, A tutorial on automatic hyperparameter tuning of deep spectral modelling for regression and classification tasks, Chemometr. Intell. Lab. Syst. 223 (2022).

[28] Colin White, Willie Neiswanger, Yash Savani, Bananas: Bayesian Optimization with Neural Architectures for Neural Architecture Search, 2019 arXiv: 1910.11858 [cs. LG].

[29] Martín Abadi, Ashish Agarwal, Barham Paul, Eugene Brevdo, Zhifeng Chen, Citro Craig, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vanhoucke Vincent, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Wattenberg Martin, Wicke Martin, Yu Yuan, Xiaoqiang Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from, tensorflow.org.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[31] Kenneth Aase Kristoffersen, Kristian Hovde Liland, Ulrike Böcker, Sileshi Gizachew Wubshet, D. Lindberg, Svein Jarle Horn, Nils Kristian Afseth, FTIR-based hierarchical modeling for prediction of average molecular weights of protein hydrolysates, Talanta 205 (2019) 12.

[32] Kristian Hovde Liland, Achim Kohler, Nils Kristian Afseth, Model-based pre-processing in Raman spectroscopy of biological samples, J. Raman Spectrosc. 47 (6) (2016).

[33] M. Graña, M.A. Veganzons, B. Ayerdi, Salinas Valley hyperspectral image, Data retrieved from Grupo de Inteligencia Computacional, https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes, https://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes.

[34] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[35] B. Walczak, D.L. Massart, The radial basis functions — partial least squares approach as a flexible non-linear regression technique, Anal. Chim. Acta 331 (3) (1996) 177–185.

[36] Rasmus Bro, Multiway calibration. multilinear pls, J. Chemometr. 10 (1) (1996) 47–61.

[37] Kristian Hovde Liland, Ulf Geir Indahl, Joakim Skogholt, Puneet Mishra, The canonical partial least squares approach to analysing multiway datasets—N-CPLS, J. Chemometr. 36 (7) (2022).

Paper IV

FUZZY REGRESSION AND CLASSIFICATION USING LOCALLY WEIGHTED ENSEMBLE MODELS (LOWEM)

Runar Helin, Ulf Geir Indahl, Oliver Tomic, Kristian Hovde Liland
Faculty of Science and Technology
Norwegian University of Life Sciences
1430 Ås, Norway

ABSTRACT

In many data collection processes, we find subgroups of observations, e.g., due to natural variation, batch effects, uncontrollable events or pooling of related observations into a larger dataset. There are various regression and classification methods in the literature trying to explicitly or implicitly model such data. We propose a flexible framework suitable for tabular data, high-dimensional data and big data with continuous and categorical responses. The concepts of locally weighted regression and Fuzzy C-regression are evolved and simplified and demonstrated on both real and simulated data. The proposed framework is shown to find meaningful subgroups and improve the prediction performance by combining local models.

***K*eywords** fuzzy clustering, local modelling, regression, classification, Perceptron

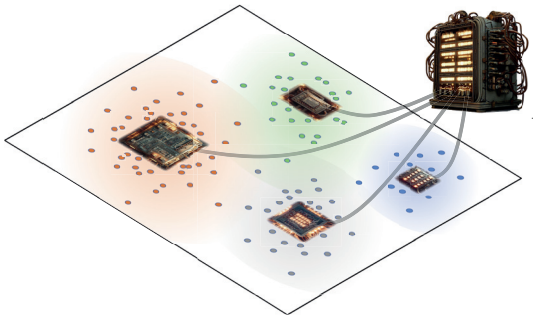


Table of Contents Figure: Illustration of a Locally Weighted Ensemble. Samples are modelled by overlapping linear models that together form a flexible ensemble.

1 Introduction

In many problems, the data collected might not be totally homogeneous. The measurements could have been taken at different time points, under different conditions, with variations in sample material or preparation, or with different equipment. Such factors might cause certain intrinsic groupings of the data. When trying to build an accurate model on such data it might be better to model each group by a local model instead of modelling all data with a global one. If the subgroups are known and recorded together with the object measurements, one can apply methods like Hot PLS [1] or Two-level PLS regression [2]. The former is a hierarchical classification method that divides the data into local subgroups before classification, while the latter classifies into known subgroups before performing local regression. Both apply Partial Least Squares (PLS) [3, 4] for modelling and dimension reduction due to high-dimensional and collinear data. If the information about such potential groupings is missing or unknown, the subgroups need to be estimated or modelled indirectly.

In the context of maximum likelihood optimisation, the expectation maximisation algorithm [5] is an efficient method when the subgroup information is missing. There are also various locally weighted regression methods, typically using kernel weighting or nearest neighbour strategies to perform local modelling and prediction [6, 7]. Another approach is to first perform an unsupervised clustering to detect possible subgroups and model these clusters locally subsequently. A possible drawback of this last approach is that there are no guarantees that the groups found by the clustering algorithm will result in a local model with low prediction error. Also, the size of the clusters may vary, which can lead to unstable models.

Tøndel et al. [8] describe a method for hierarchical cluster-based PLSR where Fuzzy C-means is applied to the scores of a global PLSR model before PLS regression is again used in the local clusters. There, the aspect of letting regression influence the choice of cluster membership is done by a single global regression model, in contrast to the iterative approach on raw input data chosen in our work.

In the literature, similar data modelling problems are sometimes termed multiple model learning [9]. The goal is to identify multiple models in a problem and includes both situations of known and unknown subgroup information. One aspect of multiple model learning is estimating the parameters (i.e., regression coefficients) of the multiple models [10]. Hathaway and Bezdek introduce the concept of Fuzzy C-regression (FCR) [11] to solve these kinds of problems. They combine fuzzy C-partitioning with the estimation of c linear model parameters and demonstrate the approach on simulated mixed data.

The FCR model is based on the assumption that the data originates from underlying models and is concerned with estimating the true parameters from the data. Since the response values are used to assign the data to the different local models, the FCR cannot be used to predict new samples where the response value is unknown. This problem was addressed by da Silva and Carvalho [12] who combined FCM with Fuzzy C-means to simultaneously perform multiple regression model estimation and fuzzy clustering.

Our study takes a similar approach. We start by introducing the framework with weighted Ordinary Least Squares (OLS) as the regression method of choice for moderately large datasets without severe multicollinearity. Except for some strategic choices, this base model coincides to a large degree with the previous work by da Silva and Carvalho [12]. As our method was originally intended for spectroscopic data with highly multicollinear features and relatively few samples, we extend the method with Partial Least Squares Regression (PLSR) as a plugin replacement for OLS. Further adjustments are also made to enable classification in the same framework. We also indicate how the framework allows for big data situations by exchanging OLS with the Perceptron [13], thus leveraging highly optimised machine learning frameworks that scale to any data size.

2 Theory and Methods

2.1 Fuzzy C-means

The goal of any clustering algorithm is to group together samples with similar attributes. In a hard-clustering algorithm such as K-means clustering [14], each sample is assigned to just one cluster. On the other hand, the Fuzzy C-means (FCM) [15] algorithm assigns a number between 0 and 1 to each sample, i, for each cluster, k, representing the membership value, μ_{ik}, to each respective cluster. Each sample is therefore a member of every cluster by an amount determined by the membership value, and $\sum_k \mu_{ik} = 1 \; \forall \; i$.

The FCM algorithm is an iterative process that starts by randomly selecting m number of cluster centres in the domain of a dataset $\boldsymbol{X}$. For each sample $\boldsymbol{x}_i$ the distance d_{ik} to each cluster centre $\boldsymbol{\theta}_k$ for $k = 1, \ldots, m$ is calculated. Based on these distances, the cluster membership values can be calculated by the equation

$$\mu_{ik} = \left[\sum_{j=1}^{m} \left(\frac{d_{ik}}{d_{ij}} \right)^{\frac{1}{q-1}} \right]^{-1}, \tag{1}$$

where q is a parameter controlling the degree of fuzziness, i.e., to which extent a sample is present in more than one cluster. In the limit where q approaches 1, the FCM algorithm is equivalent to the K-means algorithm with crisp/hard clusters.

After obtaining the cluster membership values, each cluster centre $\boldsymbol{\theta}_k$ is updated according to the standard FCM algorithm, where the cluster centres are defined by:

$$\boldsymbol{\theta}_k = \frac{\sum_{i=1}^{n} \mu_{ik}^q \boldsymbol{x}_i}{\sum_{i=1}^{n} \mu_{ik}^q}. \tag{2}$$

With the new cluster centres calculated, the cluster memberships are updated. This process is repeated until convergence, i.e., where the cluster centres stop changing or a predefined number of maximum steps is reached.

2.2 Fuzzy C-regression

A modelling approach inspired by the FCM algorithm is called Fuzzy C-regression (FCR) [11] and is an algorithm designed to estimate coefficients of multiple underlying linear models simultaneously. The main difference between FCR and FCM is that the cluster centres are defined to be regression coefficients for local linear regression models and that the distance metric d_{ij} is changed to the squared prediction error $(\boldsymbol{y} - \hat{\boldsymbol{y}})^2$.

2.3 Combining Fuzzy C-means and Fuzzy C-regression

The FCR algorithm can be successfully used to estimate model parameters for local models simultaneously. However, to predict unseen samples successfully, the underlying subgroups in the data have to be known. If this information is missing, a solution is to combine the FCM and FCR algorithms as first proposed by [12]. The authors proposed an algorithm similar to FCM but instead used a combined loss function from the FCM and FCR algorithms. With a common loss function, the cluster centres and local regression coefficients influence each other. Additionally, the authors introduce extra parameters to be optimised which correspond to the amount of relevance of explanatory variables.

2.4 The Locally Weighted Ensemble Modelling framework.

The proposed Locally Weighted Ensemble Modelling (LoWEM) framework as presented in this paper resembles the combined FCM and FCR method described above, but there are some key differences. The LoWEM framework is not only designed to be used with a basic ordinary least squares model, and it is easier to control how much the prediction error impacts the clustering. Different from [12], LoWEM calculates two cluster membership values by minimising the FCM and FCR loss functions separately and combining the results using a convex combination. More specifically, the cluster membership from the FCM algorithm is found by minimising the loss function

$$L_{FCM} = \sum_{k=1}^{m}\sum_{i=1}^{n} \mu_{ik}^{q} ||\boldsymbol{x}_i - \boldsymbol{\theta}_k||^2, \tag{3}$$

while the cluster membership for the FCR algorithm minimises the loss function

$$L_{FCR} = \sum_{k=1}^{m}\sum_{i=1}^{n} \nu_{ik}^{q} (y_i - \hat{y}_i)^2, \tag{4}$$

where the prediction error based cluster memberships, ν_{ik}, will be explained later.

These two cluster membership contributions are combined using a convex combination which preserves the property that the sum of membership values for one sample across all clusters equals 1:

$$w_{ik} = (1-\alpha)\mu_{ik} + \alpha\nu_{ik}. \tag{5}$$

Since both sets of cluster memberships scale 0-1, the α parameter is directly interpretable as balancing the contribution in $\boldsymbol{x}$-space and $\boldsymbol{y}$-space. When α is set to 0 or 1, the resulting model is equivalent to FCM and FCR, respectively.

The algorithm: Different from FCM, LoWEM initiates the m cluster centres $\boldsymbol{\theta}_k$ based on the k-means++ algorithm [16] to start with more relevant cluster centres than a random selection. The next step is to initiate the cluster membership related to prediction error ν_{ik} to 0 for every sample i and cluster k. Then, the cluster memberships based on feature similarity μ_{ik} are calculated by Equation 1, which is the solution that minimises Equation 3.

The initial LoWEM cluster membership values w_{ik} are then initialised according to Equation 5 and are used as weights when training the local models in the next step. In the original paper for the original FCR algorithm [11], this procedure is shown for OLS models. Here, the procedure is presented for PLS regression models.

The PLS algorithm tries to find a lower-dimensional representation of the data $\boldsymbol{X}$ with a set of components explaining most of the cross-correlation between $\boldsymbol{X}$ and $\boldsymbol{y}$. The result is a decomposition $\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E}$, where $\boldsymbol{T}$ and $\boldsymbol{P}$ are called scores and loadings, respectively, and $\boldsymbol{E}$ is the part of $\boldsymbol{X}$ not explained by the PLS components. The response is correspondingly decomposed as $\boldsymbol{y} = \boldsymbol{T}\boldsymbol{q}^T + \boldsymbol{F}$. These components can be found by a standard algorithm such as the NIPALS algorithm [17].

When incorporating the cluster memberships into the modelling, it is convenient to write the fuzzy membership values w_{ik}^q for each cluster k as an $n \times n$ diagonal matrix $\boldsymbol{W}_k$. Both $\boldsymbol{X}$ and $\boldsymbol{y}$ are subjected to a weighted centring by the membership values in $\boldsymbol{W}_k$. Then, scaled versions of $\boldsymbol{X}$ and $\boldsymbol{y}$ are obtained by $\tilde{\boldsymbol{X}} = \boldsymbol{W}_k^{1/2}\boldsymbol{X}$ and $\tilde{\boldsymbol{y}} = \boldsymbol{W}_k^{1/2}\boldsymbol{y}$ before passing them through the PLS algorithm. The result is a weighted PLS, where the corresponding regression coefficients in the PLS regression model incorporate the importance of the samples according to the cluster memberships. The alternative of modelling $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{y}}$ with an OLS model results in weighted least squares regression.

During the model fitting stage, the number of PLS components, which determines the model complexity, of each local PLS model is chosen based on a cross-validation procedure. The optimal number of PLS components for each local PLS is chosen as the minimum weighted cross-validated error using the corresponding cluster membership values.

After fitting each local model to the weighted samples, the cluster memberships ν_{ik} based on prediction errors r_{ik} are calculated according to

$$\nu_{ik} = \left[\sum_{j=1}^{m}\left(\frac{r_{ik}}{r_{ij}}\right)^{\frac{1}{q-1}}\right]^{-1}. \tag{6}$$

This is a solution that minimises Equation 4 and is equivalent to Equation 1 with the use of prediction errors r_{ik}, typically the squared error $r_{ik} = (y_{ik} - \hat{y}_{ik})^2$. The next step is to update the combined cluster memberships w_{ik} and

use these updated membership values to calculate new cluster centres. Here, an improved update of the cluster centres is proposed based on gradient descent. During the development of the model, it was found that including prediction error as a clustering criterion could result in many sudden large changes in cluster positions. To alleviate this effect, the following update rule for the cluster centres at iteration $p+1$ based on the current cluster centres $\boldsymbol{\theta}_k^{(p)}$ is used

$$\boldsymbol{\theta}_k^{(p+1)} = \boldsymbol{\theta}_k^{(p)} + \eta(\frac{\sum_{i=1}^{n} w_{ik}^q \boldsymbol{x}_i}{\sum_{i=1}^{n} w_{ik}^q} - \boldsymbol{\theta}_k^{(p)}), \tag{7}$$

where η is a learning rate controlling the speed of change from iteration to iteration. After obtaining new cluster centres, the process starting from fitting the local regression models repeats until convergence or a maximum number of iterations is reached.

Performing prediction using all local models

Assume the algorithm has converged and all the local models are fitted accordingly. To predict a sample $\boldsymbol{x}_i$ with corresponding cluster membership values w_{ik} for $k = 1, \dots m$, a weighted prediction of all the local models is calculated as follows

$$\hat{y}_i = \sum_{k=1}^{m} w_{ik} \boldsymbol{x}_i^T \hat{\boldsymbol{\beta}}_k \tag{8}$$

When predicting new unseen samples, the information about the prediction error is of course unknown. In that case, the cluster membership values are estimated solely based on the similarity between features according to Equation 1.

3 Extension to classification problems

The presented fuzzy clustering-based model approach can also be extended to classification. The main difference is that the output of each sample will be a vector of predictions of a one-hot encoded response. As for the regression case, the weighted prediction of the local models makes the total model capable of classifying non-linear data.

For classification, a sum of squared prediction errors can still be used to obtain the error-based cluster membership values by summing over the prediction error for all the one-hot encoded response values. For a sample i, the prediction error r_{ik} is defined by

$$r_{ik} = \sum_{c} (\boldsymbol{y}_i^{(c)} - \hat{\boldsymbol{y}}_{ik}^{(c)}), \tag{9}$$

where c denotes a specific class of the one-hot encoded response. When combining all local models for the final LoWEM prediction, the predicted one-hot prediction matrices $\hat{\boldsymbol{Y}}_k$ are combined as a weighted element-wise sum to obtain the combined prediction. To predict the categorical class, a linear discriminant analysis (LDA) can be applied using the LoWEM predictions and categorical response of the training set to fit the LDA model.

If one applies the proposed clustering algorithm choosing more clusters than classes, it is possible to perform non-linear classification that takes into account unlabelled sub-groups of one or more classes. This is especially useful in situations where one or more classes wrap (partially) around other classes in the feature space. This could for instance be caused by a too coarse level of grouping, e.g., in a biological taxonomy, leading to highly irregular group shapes in feature space.

4 Datasets

FTIR on hydrolysates
As a representative of highly heterogeneous data with multicollinear features, we have chosen the Fourier Transform Infrared (FTIR) spectra and Average Molecular Weight (AMW) measurements first described in Kristoffersen et al. [2]. These are dry-film measurements of water-soluble proteins and peptides at various time points during enzymatic protein hydrolysis reactions of various rest raw materials and commercial enzymes. In Kristoffersen et al., the 28 combinations of rest raw materials and enzymes were known and leveraged in a two-level prediction model consisting of a material/enzyme classification model and local regression models. In this work, we only use the spectra and AMW reference measurements, while the clustering will replace the known material/enzyme combinations. During the

analysis, the dataset was divided into a training and test set with 590 and 295 samples in each partition, respectively. The split was done in a stratified with respect to material/enzyme combinations and replicates were kept in the same split.

Interleaving half circles
This is a toy dataset from the software package scikit-learn [18] which is useful for illustrating the need for and application of nonlinear models in a simple two-dimensional case. The dataset contains two variables and has two classes. The total number of simulated samples in this dataset is 1000 (500 samples per class), divided into balanced training (67%) and test (33%) sets during the analysis.

FTIR on moulds
This dataset has previously been analysed using Hot PLS [1] which takes explicit advantage of the known hierarchy using a set of Canonical Powered Partial Least Squares [19] models as nodes in a tree. It further exploits replicates by employing a voting strategy in all the nodes to boost performance. The data consist of 1399 FTIR spectra in the spectral ranges 3000–2800 nm and 1800–900 nm, preprocessed with Savitzky-Golay [20] 1st order derivatives (3rd order polynomial, nine-point window) and Extended Multiplicative Signal Correction [21] with replicate correction [22] (across microtiter plate and FTIR plate replication). The moulds that were measured are associated with a phylogenetic tree with five taxonomic levels, where species (19 different) is the lowest used in classification with FTIR. Some of the mould species were represented by more than one strain. Further information is found in Liland et al. [1].

5 Results

Regression – hydrolysates: The optimal hyperparameters of the LoWEM model for the FTIR on hydrolysates dataset were found to be $\alpha = 0.1$ and $q = 1.25$, i.e., a model emphasising the X-contribution over the Y-contribution to cluster assignments and with a relatively small amount of fuzziness. The model was trained for a maximum of 50 iterations and stopped if the cluster membership norm did not change by more than 0.01. The LoWEM model was evaluated using 4, 10 and 28 clusters. For comparison, the same data was modelled by a global PLS model and a standard Convolutional Neural Network (CNN) for regression with spectral data [23]. The results are given in Table 1.

Table 1: Prediction results of the FTIR on hydrolysates dataset. Each model except the Partial Least Squares model was trained and evaluated using 10 different random states, and the standard error is given as an estimate of uncertainty.

Model	RMSEP	$R^2_{pred.}$
PLSR	414.95	0.848
LoWEM (4 clusters)	350.76 $\pm$3.14	0.892 $\pm$0.002
LoWEM (10 clusters)	322.00 $\pm$3.42	0.909 $\pm$0.002
LoWEM (28 clusters)	320.93 $\pm$9.85	0.909 $\pm$0.006
CNN	342.04 $\pm$2.8	0.897 $\pm$0.002
Two-level PLSR	281.80	0.930

We note that the global PLSR model struggles with the heterogeneity of the data caused by raw material and enzyme differences. As there were 28 known combinations of raw material and enzyme, we have used up to 28 clusters, however, we observe that the increase from 10 to 28 clusters causes more increase in instability than reduced RMSEP. From this, we concluded that the sweet spot for the number of clusters is somewhere around 10. We further note that the CNN model, which has shown promise in previous works, is not able to match the LoWEM models with more than 4 clusters. Finally, the Two-level PLSR from Kristoffersen et al. outperforms the other models by a substantial margin. In this case, it is evident that explicitly leveraging the known underlying grouping of the data in modelling is a superior choice. However, for similar data where the groups are not known, LoWEM still shows acceptable performance without the black-box properties of CNN.

When the mean membership values across samples of the same enzyme-material group are displayed as a heatmap as shown in Figure 1, it is possible to give some interpretations of the clusters in LoWEM. Immediately apparent is the fact that almost all samples of Chicken muscle hydrolysed by Papain LSG 100 (CMPa) and Chicken skin hydrolysed by Protamex (CSPr) are concentrated in just one cluster each. Furthermore, most samples of salmon (SBA, SHA, SSA) are located near cluster 2, and most mackerel (Ma, MaA, MaF, MaPa) are located near cluster 9.

For the purpose of this work, we will not go into details about what chemical properties are observed in the loading plots in Figure 3. What we do see, is the first loading of each of the four local PLS models in the four-cluster LoWEM

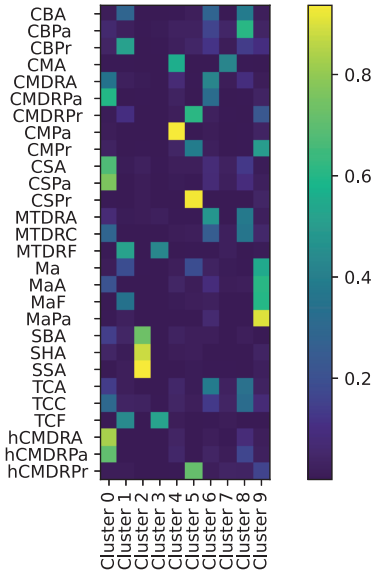


Figure 1: Mean cluster weight per cluster for each enzyme-material combination in the hydrolysis training set.

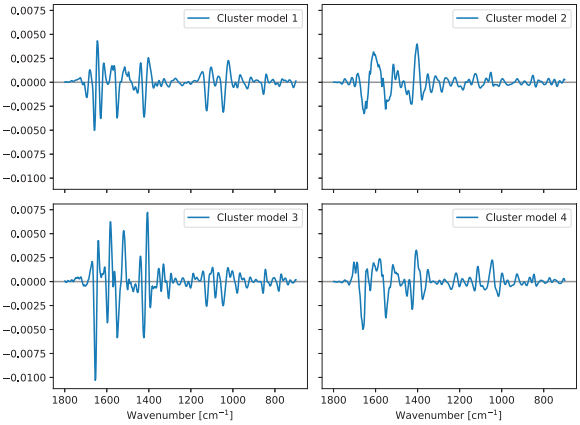
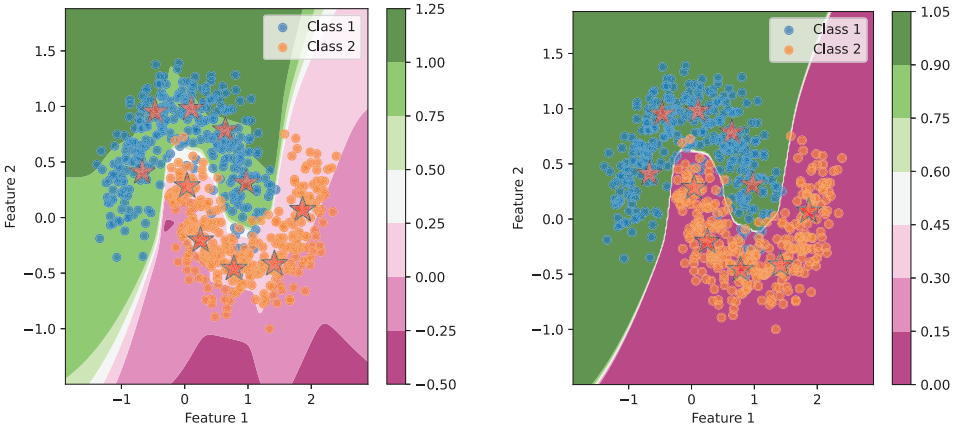


Figure 2: Loadings plot of the first PLS component for each local PLS model in LoWEM using 4 clusters.

on the hydrolysis data. It is evident that the chemical focus is quite different between the models, especially when looking at clusters 2 and 4.

Classification – half circles: The separating half circles data is an example of data that are not linearly separable, which is evident from the accuracy of the PLS-DA model in Table 2. A good choice of hyperparameters for the LoWEM on this dataset was found to be $\alpha = 0.1$ and $q = 1.1$. Figure 3 shows the results from the LoWEM modelling when using 10 cluster centres. The cluster centres, marked with stars, get distributed among the data with high density. The background colour shows the model prediction at different points in the feature space and gives a decision boundary for the classification problem. The LDA decision boundary is very crisp indicating a confident model. Each local model is



(a) LoWEM prediction

(b) LoWEM prediction after linear discriminant analysis (LDA)

Figure 3: Intersection half circles with two classes and ten local linear models. The background colour indicates the confidence of classification to Class 1 (green) and Class 2 (purple) given the LoWEM model.

linear and can only create a linear decision boundary. However, when combining all models the decision boundary becomes more complex and able to separate the two classes more effectively.

Table 2: Prediction accuracy of the simulated classification dataset. Each model has been trained and evaluated using 10 different random states, and the standard error is given as estimate uncertainty.

Model	Accuracy
LDA	88.2% $\pm$0.0
LoWEM (2 clusters)	92.8% $\pm$0.05
LoWEM (5 clusters)	95.4% $\pm$0.05
LoWEM (10 clusters)	96.3% $\pm$0.04
Random Forest	95.7% $\pm$0.05

In the case of these simple data, we see that the number of clusters can be quite high without incurring a penalty in the form of higher uncertainty on the accuracy. We also see that, without fine-tuning of either model, LoWEM is able to beat the accuracy of Random Forest. Looking at the overlap between the two classes in Figure 3, it is not likely that performance can be pushed much higher than what has been achieved without sacrificing generalisability.

Classification – moulds: Knowing that there are two *divisions*, five *classes*, nine *genera* and 11 *sub-genera* that the 19 *species* are nested under, the choice of number of clusters is not given in advance. In addition, the previous example clearly showed that more than one cluster per class can be useful for highly non-linear class boundaries. As for the hydrolysis data, we observe a substantial improvement from using a global PLS-DA model. In this case, two out of six biological replicates also show higher accuracy for LoWEM than the highly tailored Hot PLS method (explicit use of phylogeny and voting between replicates). As seen for the hydrolysates data, the uncertainties of the predictions increase somewhat when increasing the number of clusters. A good choice of LoWEM hyperparameters for this dataset was $\alpha = 0.1$ and $q = 1.25$, the same as for the hydrolysis data.

The bar plots in Figure 4 are generated by counting how many of the training samples are closest to each of the clusters when the 6th biological replicate is held out for classification. The first thing we observe is that clusters 2, 6 and 7 are dominated by a single species for the case with 10 clusters. There is also a high degree of phylogenetic similarity between species that share a cluster as can be confirmed by comparing to the phylogenetic tree in [1]. For instance, all three species of the Mucor genus (M.) and the related Rhizopus genus (R.) are present in clusters 3 and 8, all three species of the Aspergillus genus (A.) are present in cluster 9, and so on. Similar information is presented in Figure 5 where the mean membership values across samples of the same species are displayed as a heat map for the case with 10 clusters. Here we observe that most of the species have the bulk of their membership values spread over one or two

Table 3: FTIR on moulds prediction accuracies. The overall errors across all biological replicates are shown in the first row. The results from [1] are included as a reference. The standard error of 10 random initialisations of LoWEM are included.

Biological replicate	PLS-DA	LoWEM (2 clusters)	LoWEM (10 clusters)	LoWEM (19 clusters)	Hot PLS
All combined	81.0%	77.0%±0.09	86.0%±0.17	87.7±0.29	87.7%
1st	85.5%	80.9%±0.19	86.1%±0.25	85.7±0.76	86.0%
2nd	77.8%	75.5%±0.20	82.3%±0.47	87.7±0.79	82.6%
3rd	88.1%	78.8%±0.63	92.9%±0.43	96.5±0.43	93.2%
4th	66.0%	66.4%±0.17	72.9%±0.77	73.3±0.73	76.2%
5th	84.2%	80.5%±0.13	94.5%±0.59	95.2±0.28	98.3%
6th	84.7%	79.7%±0.15	87.4%±0.57	88.0±0.62	89.8%

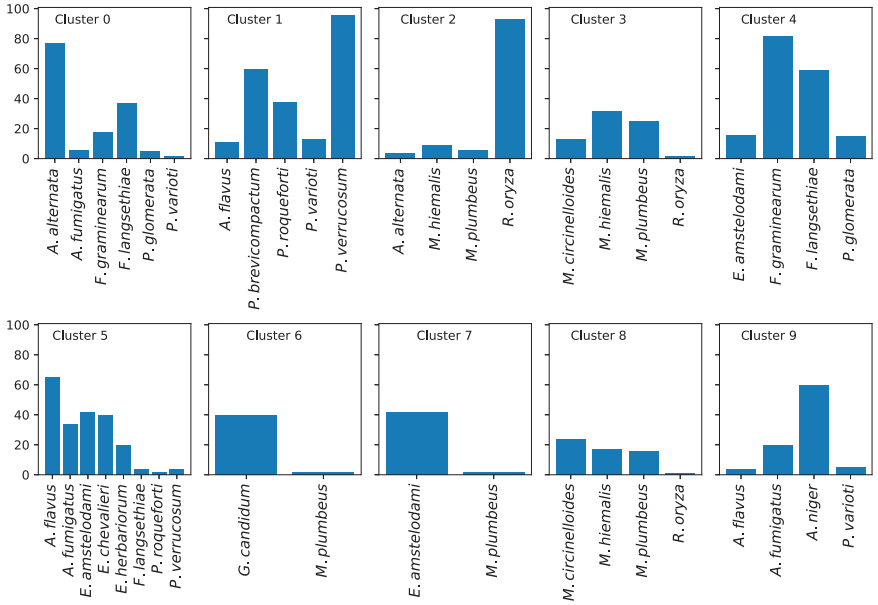


Figure 4: Histogram of species for samples closest to each cluster center

clusters, while a few are spread over three, e.g., two of the Mucors (M.). These results indicate that the cluster centres themselves tend to converge to clusters of samples having some biological similarity.

6 Discussion

Linear models and parsimonious data

Artificial neural networks are making a huge impact in the fields of image analysis, language modelling and other parts of artificial intelligence, but are also finding use in regression, classification and spectroscopic data analysis. In this work, we have continued and expanded the work on local linear models as a transparent and flexible alternative to the more or less black-box approach with complex neural networks. As demonstrated in the examples, linear models like PLS can model highly heterogeneous data when applied smartly while still giving linear views of the underlying phenomena by low-dimensional compressions. This means we can push the envelope for visualisation and interpretation further while handling more complex problems with local and non-linear phenomena. OLS/LDA and PLS are also proven to work well in situations with short wide datasets where the number of features greatly outnumbers the number of samples.

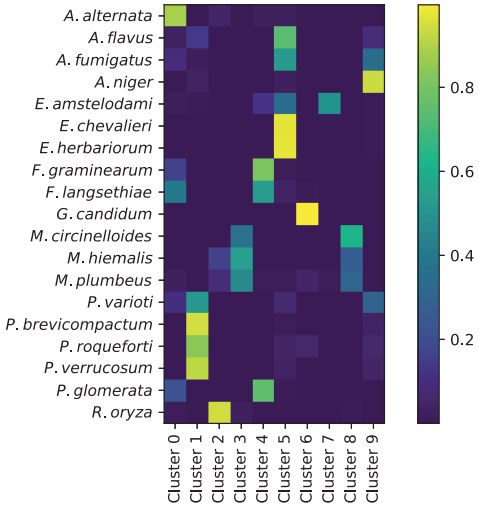


Figure 5: Mean cluster weight per cluster for each species in the training set.

Reduced complexities in local models
In addition to increasing the overall prediction performance compared to a single global PLS model, it was observed that each local PLS model in LoWEM needed fewer PLS components to achieve these results. The global PLS model presumably needed more components to compensate for the different underlying groupings in the data. In general, it is more desirable to have models with fever components since the later components contain more noise.

When modelling new data, each model will likely be dominated by subspace structures similar to a global PLS. A good starting point for tuning individual models is therefore to set a maximum number of components equal to or slightly higher than the optimum for a global PLS model (to allow for individual differences) and let cluster-wise cross-validation decide the final number of components. This may seem like an excessive complexity for the full LoWEM model, however, since the local models likely share overlapping properties, this should not be a concern.

Hyperparameter tuning
For the OLS/LDA-based LoWEM there are three parameters to be tuned: (1) the number of clusters (m), (2) the degree of fuzziness (q), and (3) the balance between $\boldsymbol{x}$-space and $\boldsymbol{y}$-space in membership updates (α). Exchanging OLS/LDA with PLSR/PLS-DA, a fourth parameter is the number of components. In addition, it is possible to use a learning rate for smoother transitions from iteration to iteration. Compared to a single global model, this is indeed more complicated, and some parameters may even have interactions with each other.

Our experience with the three datasets in this work indicates that a relatively crisp clustering with $q \approx 1.1 - 1.25$ was a good choice. The relatively crisp clustering gave the best prediction performance. However, the crisp clusters also resulted in large jumps in cluster membership values when samples "changed" clusters between iterations, resulting in poor convergence. The challenge of unstable model convergence was addressed by basing the modelling on a fuzzy clustering approach [15]. By increasing the fuzziness, smoother convergence is obtained but at a possible sacrifice of some prediction performance. The number of clusters could be chosen surprisingly high, even for classification problems. However, for both the FTIR-based datasets we observed less stable solutions when the number of clusters was high than for more parsimonious solutions. There is also a balance between prediction performance and computational cost, as it was observed that more clusters tend to give better predictions but require more computations. The balance between $\boldsymbol{x}$-space and $\boldsymbol{y}$-space was best at $\alpha \approx 0.1$ for these datasets. Increasing the value, i.e., giving more weight to predictions, means that the basis for assigning new samples to existing clusters becomes weaker while moving too close to $\alpha = 0$ would defeat the purpose of including the success of regression in the cluster membership criterion. Finally, the number of components in PLS was tuned using cross-validation for each of the clusters, so only an upper bound is needed. Our recommendation is to first optimise this on a global model and use the optimal number as a starting point in the local models, possibly increasing the maximum with a few components to allow for locally more complex models.

Challenges and limitations
The main challenges with the LoWEM framework compared to a single global model are the added complexity of having more parameters to tune and the time it takes to fit the model. Since there are several local models and these are iteratively refitted, the total number of computations will be higher than for a global model. Using an efficient PLS algorithm is important. For optimising the number of components, one can save time by only cross-validating every n-th model and otherwise reusing the number of components. This could, however, affect the convergence of the cluster centres. It is also possible to exchange cross-validation with a different procedure, e.g., applying an efficient weight randomisation test [24] as was done with success on spectral data in Mishra et al. [25]. Concerning the other hyperparameters of LoWEM a strategy of coarse search followed by fine-tuning is recommended.

Upscaling to big data
An alternative to using OLS or PLS is the use of the linear Perceptron model. Though the optimisation criterion for the Perceptron is different from OLS, its solution is typically very similar to that of OLS. This opens up the possibility to train every model batch-wise, tackling problems with huge datasets and also problems with continuous streams of data. Libraries such as TensorFlow or Pytorch offer quick access to efficient implementations of the Perceptron model, ready to be employed on high-performance computing (HPC) clusters. As the use of the Perceptron is a direct plugin replacement for OLS, we will not be demonstrating its use in this work.

Comparison with neural networks
There is an interesting comparison between the multiple modelling aspect presented here and neural networks. Recently, it has been suggested that a large group of artificial neural network (ANN) models, including CNNs, can be written as a composition of max-affine spline operators (MASOs) [26]. Among the results is the suggestion that a MASO ANN is performing a hierarchical, greedy template matching on the input signal. The authors discuss these results for image classification problems, but the result should be analogous for regression problems. In other words, a MASO ANN is comparing input signals to learned "prototypes" of data and makes predictions based on closeness to them. This concept sounds similar to the LoWEM approach presented in our paper. If the MASO ANN theory is correct, it gives hope of being able to achieve similar prediction performance as these types of ANN models with other modelling approaches. The results obtained in this paper regarding the hydrolysates data support this claim.

Future work
We have already mentioned some of the possible steps further like alternative ways of estimating the number of components in the PLS models and using a learning rate to smooth transitions from iteration to iteration. The latter can be further expanded on by setting up a learning rate schedule that lets the model make large changes in cluster memberships early in the modelling and then reduce the learning rate as the model moves towards completion. One could also envision a hybrid approach where known subgroups are used as a seed for a cluster-based approach where the subgroups are allowed to evolve into something more useful for minimising prediction error than blinding trusting the known subgroups that may be only partially relevant.

Another possible step is to implement efficient parallelisation to LoWEM. Since each local model training and cross-validation is independent within each iteration, it is possible to distribute these computations across multiple processors.

7 Conclusion

We have introduced a framework for fuzzy regression and classification using locally weighted ensemble models and demonstrated how it handles heterogeneous data well without prior knowledge of subgroups. This is an evolution of previous work where balancing the focus on $\boldsymbol{x}$-space and $\boldsymbol{y}$-space is better described, classification is included in the framework and expansion to big data is trivial. Transparent modelling of complex data comes at the cost of extra hyperparameters to tune and higher computational cost, however, we avoid black-box modelling and thousands of parameters to estimate.

References

[1] Kristian Hovde Liland, Achim Kohler, and Volha Shapaval. Hot pls—a framework for hierarchically ordered taxonomic classification by partial least squares. *Chemometrics and Intelligent Laboratory Systems*, 138:41–47, 2014.

[2] Kenneth Aase Kristoffersen, Kristian Hovde Liland, Ulrike Böcker, Sileshi Gizachew Wubshet, Diana Lindberg, Svein Jarle Horn, and Nils Kristian Afseth. Ftir-based hierarchical modeling for prediction of average molecular weights of protein hydrolysates. *Talanta*, 205:120084, 2019.

[3] Svante Wold, Harold Martens, and Herman Wold. The multivariate calibration problem in chemistry solved by the PLS method. In *Matrix pencils*, pages 286–293. Springer, 1983.

[4] S Wold, A Ruhe, H Wold, and W J Dunn. The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. *SIAM Journal of Scientific and Statistical Computing*, 5:735–743, 1984.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39:1–22, 1977.

[6] William S Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[7] Tormod Naes, Tomas Isaksson, and Bruce Kowalski. Locally weighted regression and scatter correction for near-infrared reflectance data. *Analytical Chemistry*, 62(7):664–673, 1990.

[8] Kristin Tøndel, Ulf G Indahl, Arne B Gjuvsland, Jon Olav Vik, Peter Hunter, Stig W Omholt, and Harald Martens. Hierarchical cluster-based partial least squares regression (hc-plsr) is an efficient tool for metamodelling of nonlinear dynamic models. *BMC Systems Biology*, 5(1):1–17, 2011.

[9] Vidyashankar Kuppuraj and Raghunathan Rengaswamy. Evaluation of prediction error based fuzzy model clustering approaches for multiple model learning. *International Journal of Advances in Engineering Sciences and Applied Mathematics*, 4:10–21, 2012.

[10] V. Cherkassky and Y. Ma. Multiple model regression estimation. *IEEE Transactions on Neural Networks*, 16:785–798, 2005.

[11] Richard J Hathaway and James C Bezdek. Switching regression models and fuzzy clustering. *IEEE Transactions on fuzzy systems*, 1(3):195–204, 1993.

[12] Ricardo AM da Silva and Francisco de AT de Carvalho. On combining fuzzy c-regression models and fuzzy c-means with automated weighting of the explanatory variables. In *2018 IEEE international conference on fuzzy systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2018.

[13] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[14] Hugo Steinhaus et al. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci*, 1(804):801, 1956.

[15] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Plenum, 1981.

[16] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[17] Herman Wold. Path models with latent variables: The nipals approach. In *Quantitative sociology*, pages 307–357. Elsevier, 1975.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] Ulf G Indahl, Kristian Hovde Liland, and Tormod Næs. Canonical partial least squares—a unified pls approach to classification and regression problems. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 23(9):495–504, 2009.

[20] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[21] Harald Martens and Edward Stark. Extended multiplicative signal correction and spectral interference subtraction: new preprocessing methods for near infrared spectroscopy. *Journal of pharmaceutical and biomedical analysis*, 9(8):625–635, 1991.

[22] A Kohler, U Böcker, J Warringer, A Blomberg, SW Omholt, E Stark, and H Martens. Reducing inter-replicate variation in fourier transform infrared spectroscopy by extended multiplicative signal correction. *Applied spectroscopy*, 63(3):296–305, 2009.

[23] Runar Helin, Ulf Geir Indahl, Oliver Tomic, and Kristian Hovde Liland. Non-linear shrinking of linear model errors. *Analytica Chimica Acta*, in press, 2023.

[24] Thanh Tran, Ewa Szymańska, Jan Gerretzen, Lutgarde Buydens, Nelson Lee Afanador, and Lionel Blanchet. Weight randomization test for the selection of the number of components in pls models. *Journal of Chemometrics*, 31(5):e2887, 2017.

[25] Puneet Mishra, Junli Xu, Kristian Hovde Liland, and Thanh Tran. Meta-pls modelling: An integrated approach to automatic model optimization for near-infrared spectra. *Analytica Chimica Acta*, 1221:340142, 2022.

[26] Randall Balestriero and richard baraniuk. A spline theory of deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 374–383. PMLR, 2018.

ISBN: 978-82-575-2076-2
ISSN: 1894-6402

Norwegian University
of Life Sciences

Postboks 5003
NO-1432 Ås, Norway
+47 67 23 00 00
www.nmbu.no

117064 / ANDVORDGRAFISK.NO