Peer Reviewed Tutorial openaccess

Deep learning classifiers for near infrared spectral imaging: a tutorial

Jun-Li Xu,[a,*] Cecilia Riccioli,[b] Ana Herrero-Langreo[a] and Aoife A. Gowen[a]

[a]UCD School of Biosystems and Food Engineering, University College of Dublin (UCD), Belfield, Dublin 4, Ireland

[b]Faculty of Agriculture and Forestry Engineering, Department of Animal Production, University of Cordoba, Cordoba, Spain

Contacts

J.-L. Xu: junli.xu@ucd.ie; https://orcid.org/0000-0002-4442-7538
C. Riccioli: https://orcid.org/0000-0002-0998-7150
A. Herrero-Langreo: https://orcid.org/0000-0003-3258-6248
A.A. Gowen: https://orcid.org/0000-0002-9494-2204

Deep learning (DL) has recently achieved considerable successes in a wide range of applications, such as speech recognition, machine translation and visual recognition. This tutorial provides guidelines and useful strategies to apply DL techniques to address pixel-wise classification of spectral images. A one-dimensional convolutional neural network (1-D CNN) is used to extract features from the spectral domain, which are subsequently used for classification. In contrast to conventional classification methods for spectral images that examine primarily the spectral context, a three-dimensional (3-D) CNN is applied to simultaneously extract spatial and spectral features to enhance classification accuracy. This tutorial paper explains, in a stepwise manner, how to develop 1-D CNN and 3-D CNN models to discriminate spectral imaging data in a food authenticity context. The example image data provided consists of three varieties of puffed cereals imaged in the NIR range (943–1643 nm). The tutorial is presented in the MATLAB environment and scripts and dataset used are provided. Starting from spectral image pre-processing (background removal and spectral pre-treatment), the typical steps encountered in development of CNN models are presented. The example dataset provided demonstrates that deep learning approaches can increase classification accuracy compared to conventional approaches, increasing the accuracy of the model tested on an independent image from 92.33 % using partial least squares-discriminant analysis to 99.4 % using 3-CNN model at pixel level. The paper concludes with a discussion on the challenges and suggestions in the application of DL techniques for spectral image classification.

Keywords: spectral imaging, deep learning, near infrared, classification, convolutional neural network

Introduction

As an integration of spectroscopy and digital imaging techniques, spectral imaging has emerged as a versatile tool for many applications including remote sensing,[1,2] food sciences,[3,4] pharmaceutical research,[5,6] forensic sciences,[7,8] cultural heritage,[9] agriculture and forestry.[10,11] A spectral image is a three-dimensional (3-D) data array with two spatial dimensions (of x rows and y columns) and one spectral dimension (of λ wavelengths). A spectral image, hereafter denoted $I(x,y,\lambda)$, can be visualised either as an intensity image $I(x,y)$ at each wavelength λ, or as a spectrum $I(\lambda)$ at each pixel (x,y). The spectrum, which can be obtained by plotting the absorbance/reflectance/transmittance as a function of wavelength, enables quantification or

Correspondence
Jun-Li Xu (junli.xu@ucd.ie)

Received: 14 September 2020
Revised: 4 December 2020
Accepted: 21 December 2020
Publication: 24 December 2020
doi: 10.1255/jsi.2020.a19
ISSN: 2040-4565

Citation
J.-L. Xu, C. Riccioli, A. Herrero-Langreo and A.A. Gowen, "Deep learning classifiers for near infrared spectral imaging: a tutorial", *J. Spectral Imaging* **9**, a19 (2020). https://doi.org/10.1255/jsi.2020.a19

classification of material(s) within an image at the individual pixel level.[12]

A general, though not exclusive, scheme to perform pixel-wise classification of a spectral imaging dataset includes: 1) unfolding (i.e., converting 3-D spectral image into 2-D matrix); 2) spectral pre-processing; 3) model development; and 4) re-folding (i.e., transforming the predicted values of every pixel into a classification map). Additional steps, such as background removal, region of interest (ROI) selection and image processing, can also be carried out depending on the spectral image composition. Partial least squares-discriminant analysis (PLS-DA)[13] is a supervised class-modelling method that uses a PLS algorithm (e.g. non-iterative partial least squares[14]) to predict the membership of a sample or spectrum in a given class. PLS-DA is often used in spectral data analysis for classification problems due to its capability to deal with the multicollinearity problem in near infrared (NIR) spectra which occurs because of very high intercorrelation between measured absorbances at consecutive wavelengths.[15,16] Machine learning (ML) techniques have also been proposed for spectral imaging data classification, and are prominent in the analysis of remotely sensed data.[17] A revolution in the ML field has occurred recently due to the establishment of deep learning (DL) models.[18] These models have enabled the development of new and enhanced spectral imaging data classifiers.[17–19] Convolutional neural networks (CNN), at the forefront of the current state-of-the-art in deep learning,[20] first achieved successes in the field of image recognition and have become an increasingly popular tool for remotely sensed spectral imaging data classification.[17]

In this context, we illustrate the application of 1-D CNN and 3-D-CNN models for spectral image classification. Example dataset and MATLAB scripts are freely available to be downloaded from https://bitbucket.org/lily-xu/deep-learning-classifiers-for-near-infrared-spectral-imaging/downloads/. All data analysis was conducted in the MATLAB computing environment (release R2019a, The MathWorks, Inc., Natick, MA, USA) incorporating functions from Deep Learning Toolbox, Statistics and Machine Learning Toolbox, Image Processing Toolbox and additional functions written in-house, such as pre-processing and PLS-DA and decision tree (Dtree) modelling. It is important to note that all MATLAB scripts provided should be followed sequentially, in line with the structure of the tutorial, i.e. every command line depends upon previous lines being executed. Full understanding of this tutorial requires basic knowledge of spectral imaging analysis in general and MATLAB programming. It is hoped that this paper will provide a basic understanding of the use of DL techniques for spectral imaging classification and encourage the adaptation of some useful strategies to solve individual problems.

Prior to classification

The data structure of a spectral image consists of many pixel spectra collected from the measured area. As a result, it is important to understand the structure of a given dataset and then select the appropriate tools to deal with the final data processing objective. This section will introduce the structure of the example dataset and essential procedures (i.e. background removal and spectral pre-processing) to conduct prior to classification task. The corresponding MATLAB script is called "Section2_Prior_to_Classification.m".

Spectral imaging instrumentation

Spectral images of the cereal samples were collected by a laboratory-based pushbroom spectral imaging system. This system consists of the following main components: an imaging spectrograph (Specim N17E, Spectral Imaging Ltd., Oulu, Finland) and an InGaAs camera (InGAs 12-bit SU320MS-1.7RT Sensors Unlimited, Inc.). The reflectance image is acquired in the spectral range of 880–1720 nm with an interval of 7 nm. Since the beginning and end spectral regions suffer from significant noise, only spectral data in the 943–1643 nm range are retained leading to 101 spectral bands. Direct reflectance spectra are used for subsequent data analysis.

Spectral imaging dataset

A similar cereal dataset from Gowen *et al.*[21] is used in this work to evaluate classification performance using DL. It consists of NIR spectral images of three types of puffed cereals: honey nut cornflakes, crunchy cookie cereal and crisp flakes of rice made and purchased from Tesco in Ireland. These samples are of great interest because they have image texture and spectral difference among classes, while the spatial inhomogeneity due to sample morphology could pose some challenges in classification. Samples are labelled as Corn, Wheat and Rice, respectively, according to their main components. The representative colour images of them, which were captured by a computer vision system as described by Xu and Sun,[22] are shown in Figure 1. Four spectral images of each sample type were obtained. Figure 1 shows the mean image of the spectral domain, which is computed by averaging the reflectance spectrum of each pixel. As seen, the first three

samples of each type are selected as the training set for model development, while the remaining one serves as the validation set, leading to nine spectral images consisting of 14,589 pixels for training and three spectral images of 4967 pixels for validation. The structure array data type in MATLAB enables grouping and saving various data types using a data container which is known as field. In this work, the structure array data type is used to store these 12 spectral images and named "Cereal". In addition to this, the developed models were tested on a "mixed image" containing three samples of each material. It should be noted that samples from the mixed image are not used in the training set, making them suitable to form an independent test set. The mean image and ground truth of this mixture are shown in Figure S1 of the supplementary material.

Background removal

Samples were placed on a white tile for imaging. Thresholding was applied to remove image background. The reflectance image at 1496 nm was subtracted from that of 1125 nm to enhance the contrast between sample and white tile. Afterwards, a threshold of 0.2 is applied for background removal.

Spectral pre-processing

Spectral pre-processing strategies are carried out to avoid the influence of unwanted phenomena originating from the measurement or sample properties (such as light scattering). In this work, two classical spectral pre-processing methods are employed, namely standard normal variate (SNV) and first derivative. SNV is usually used to alleviate multiplicative interferences by subtracting the mean value from each spectrum and subsequently dividing by the standard deviation.[23] First derivative is performed using Savitzky–Golay (SG) filtering[24] with a window size of 11 and a third order polynomial degree.

Figure 2 shows the mean spectra of all objects (i.e., three objects for each category) in the training set and the outcome of pre-treatments. As can be seen, it is challenging to discriminate among three cereals on the basis of the raw spectra (Figure 2A). As shown in Figure 2B, the spectral difference between the different types of cereals at 970 nm, which might be attributed to second overtone of O–H stretching from water according to Cheng and Sun,[25] becomes more prominent after the scattering is reduced by applying SNV. The baseline is largely reduced after performing first derivative (Figure 2C). Discrimination among cereals is improved, as witnessed by the difference at some wavelengths such as 1420 nm which could be assigned to the first overtone of O–H stretching.[26] The combination of SNV and first derivative is also applied, as shown in Figure 2D. Compared to preforming first derivative alone, the combination of pre-treatments

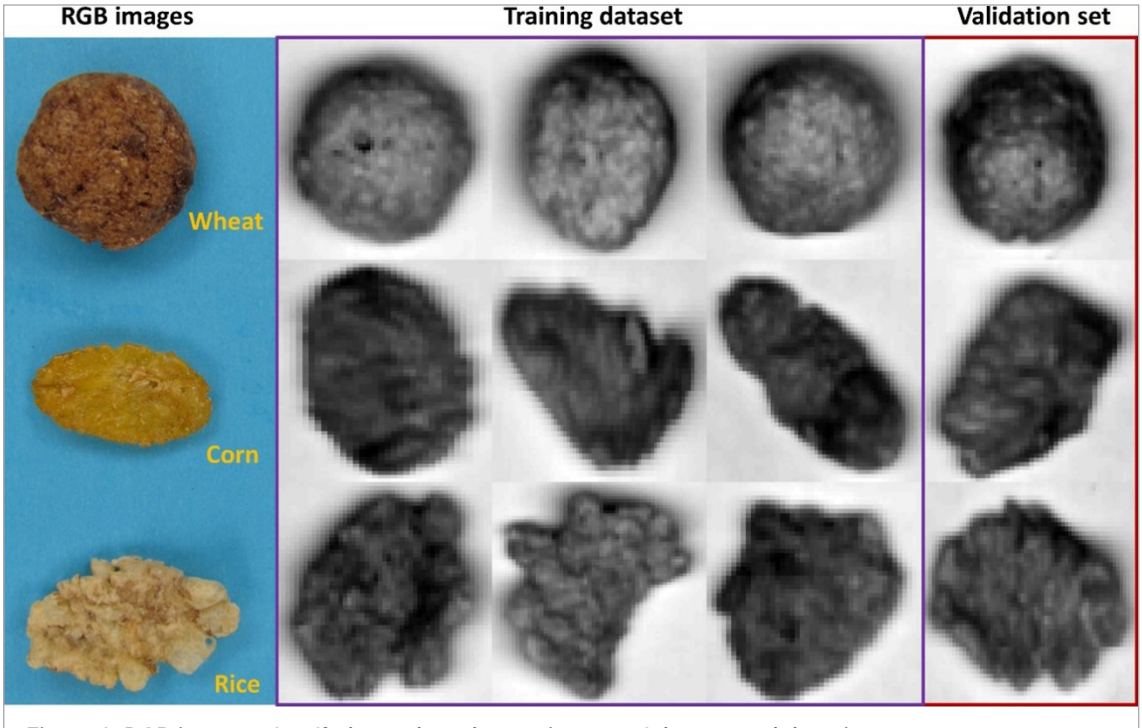

Figure 1. RGB images of puffed cereals and mean images of the spectral domain.

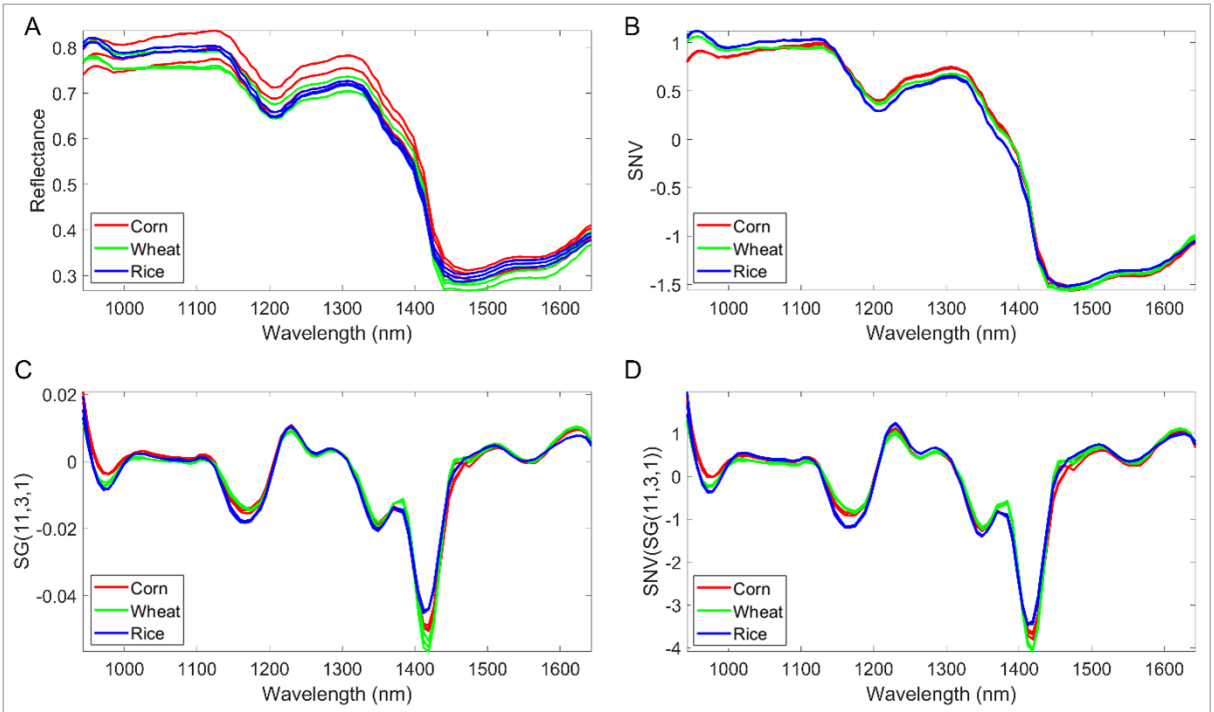

Figure 2. Mean spectra of each spectral image included in the training set before and after different pre-processing techniques. (A) Raw reflectance spectra; (B) SNV pre-processed spectra; (C) first derivative pre-processed spectra (SG, window size = 11, polynomial degree = 3); (D) first derivative pre-processed spectra followed by SNV.

enables the reduction of variations within the same category, which is beneficial for the subsequent classification task.

Assessment of classification models

Classification models were built on a training set (comprising pixel spectra extracted from three spectral images of each material) and applied on a validation set (comprising pixel spectra extracted from one image of each material), to enable comparison of model performance. In addition, a test set, comprising a mixture of all classes is used for model evaluation. Primarily, the performance of the developed model is assessed by the classification accuracy, i.e., % correct classification rate (%CCR). The confusion matrix is used to evaluate the classifier performance for validation and test sets (see an example of Figure 4A). We also calculate the percentages of samples belonging to each class that are correctly and incorrectly classified, as shown on the far right of each confusion matrix. The row at the bottom of confusion matrix shows the percentages of all the observations predicted to each class that are correctly classified and misclassified. The sensitivity [or true positive rate (TPR)], false negative rate (FNR), precision [or positive predictive value (PPV)] and false discovery rate (FDR), are computed as shown below in Equations 1–4, where TP and TN refer to true positive and true negative, respectively [i.e. observations correctly predicted as belonging (TP), or not belonging (TN), to a specific class]. FP and FN refer to false positive and false negative, respectively, [i.e. observations incorrectly predicted as belonging (FP), or not belonging (FN), to a specific class].

$$TPR = \frac{TP}{TP + FN} \quad (1)$$

$$FNR = \frac{FN}{FN + TP} \quad (2)$$

$$PPV = \frac{TP}{TP + FP} \quad (3)$$

$$FDR = \frac{FP}{FP + TN} \quad (4)$$

Superior classification performance is characterised with higher CCR, TPR, PPV and lower FNR and FDR. Additionally, classification and misclassification maps are

also presented to visualise the locations of correctly and incorrectly classified pixels, respectively.

PLS-DA and Dtree modelling

Model development

For comparison, partial least squares-discriminant analysis (PLS-DA)[27] and a decision tree (Dtree) classifier are used to build classification models. PLS-DA is one of the most popular classification methods for spectral imaging datasets,[28] therefore, it is chosen to compare with DL classifiers. As a non-linear classifier, decision tree classification (Dtree) is selected—this is among the most popular machine learning algorithms. Sample pixels from each spectral image are extracted by unfolding and concatenated to make a two-dimensional matrix (**X**). PLS-DA and Dtree models were developed based on **X** and **Y** (i.e., a matrix where the rows represent observations and columns represent the true classes), the whole procedure including model assessment is provided in the script "Section4_PLS-DA_Dtree_Classification.m".

It is important to select the appropriate number of latent variables (LVs) for a PLS-DA model. Selection of too few or too many LVs are both unsatisfactory, since either approach will result in, respectively, under or over-fitting of the data, both of which result in poor model performance.[29] In this work, venetian blinds cross-validation is applied to determine the optimal number of LVs, which is performed by checking the evolution of the CCR with the number of LVs, as illustrated in Figure S2 produced by running the function of "Nplsda_CV.m". The accuracy increases rapidly for the first few LVs and then remains relatively constant, i.e., including more variables will not enhance accuracy.

Classification performance

Classification model performance of PLS-DA in terms of CCR (%) calculated on the validation set and test image are shown in Table 1. Overall, the prediction results for test set (i.e., prediction of mixture image) are inferior to that of validation set. This is probably because the test set mixture image has three objects for each sample type (3108 pixels for Corn, 5742 pixels for Wheat and 4706 pixels for Rice), while the validation set only contains one object of each type (1224 pixels for Corn, 1987 pixels for Wheat and 1756 pixels for Rice). In this sense, comparison based on the test set is more appropriate due to its larger dataset. It is notable that pre-treatments have improved accuracy for the test set, from CCR of 89.23% for raw spectra to CCR of 91.56% for SNV combined with first derivative pre-processed spectra.

Table 1. PLS-DA classification model performance for validation and prediction image (i.e., test set) in terms of % correct classification rate (%CCR).

Pre-treatments	LVs	Validation	Test
Raw	10	93.82	89.23
SNV	10	93.50	89.41
SG	7	91.99	88.06
SG+SNV	8	92.85	91.56

Note: SNV: standard normal variate; SG: first derivative using Saviztky–Golay; LVs: latent variables.

In addition, confusion matrices for validation and test sets are displayed in Figure S3 and Figure S4 of the supplementary material, respectively. As illustrated, classification of wheat pixels has higher TPR (i.e., sensitivity) and PPV (i.e., precision), corresponding to Figure 2C and D where wheat presents distinctive spectral features covering 1350–1450 nm. We can also tell from the confusion matrix that a high number of rice pixels are incorrectly classified as corn. In order to produce a classification map, the mixture spectral image is first unfolded with background pixels removed using masking to form a two-dimensional matrix on which the developed classifier can be applied. Finally, the resultant matrix with predicted class assigned to each pixel needs to be refolded to generate a classification map, as shown in Figure 3. Many rice pixels are incorrectly predicted as corn, which corresponds to the confusion matrix. It also can be noticed that some misclassified pixels are distributed around the edge.

The model performance of Dtree classifier built from first derivative transformation followed by SNV pre-processing is shown in Figure 4. The Dtree classifier produces slightly better predictive ability than PLS-DA, e.g., the CCR for test set is 91.56% for PLS-DA while the CCR increases to 92.79% for Dtree.

One-dimensional CNN modelling

Architecture of 1-D CNN

A special case of CNN, the 1-D CNN, can be applied to one-dimensional data, such as spectroscopic data.[30] The architecture of the 1-D CNN comprises an input layer,

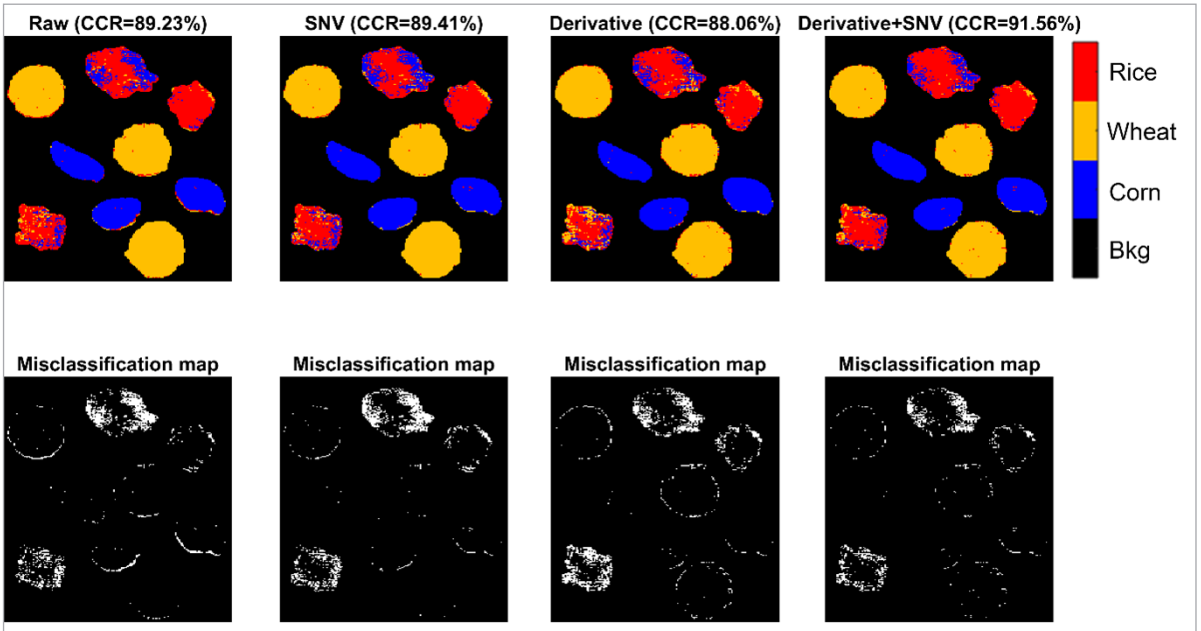

Figure 3. Classification maps (top row) and misclassification maps (bottom row) for PLS-DA models built with raw spectra, SNV pre-treated spectra, first derivative pre-treated spectra and first derivative followed by SNV pre-treated spectra.

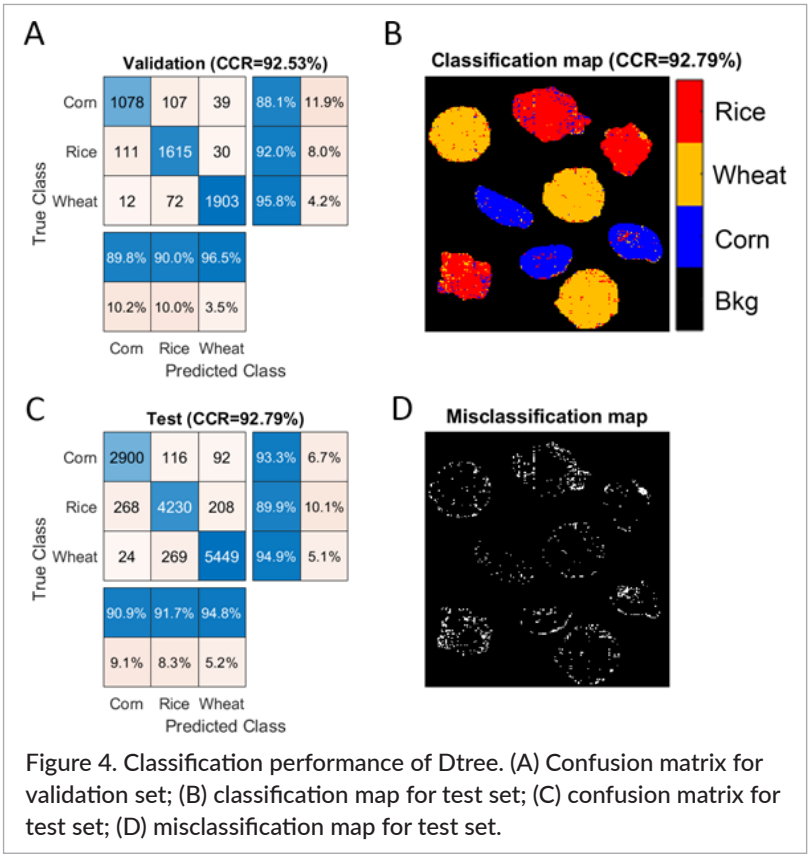

Figure 4. Classification performance of Dtree. (A) Confusion matrix for validation set; (B) classification map for test set; (C) confusion matrix for test set; (D) misclassification map for test set.

a convolution (Conv) layer, a batch normalisation (BN) layer, a rectified linear unit (ReLU) layer, a dropout layer, a fully connected (FC) layer, a softmax layer and an output layer, as shown in Figure 5. Each layer is described in more detail below.

The input of the 1-D CNN is a spectrum, i.e. a one-dimensional vector with the size of 1×101 (i.e. spectral bands) in this example dataset, therefore, the first step is to extract the spectral vector from the 3-D spectral image. This can be done as described in the Model

development section. The function of the convolutional layer is to convolve the input data by applying sliding convolutional filters and produces the convolved features as the output[31] also known as feature maps. In other words, each type of extracted feature is generated by a convolutional kernel. Conventionally, the kernel is moved first from left to right and then from top to bottom over the input with a step of 1. Strided convolution has a larger user-defined step size for traversing the input. For a 1-D CNN, the convolution kernel (also known as a filter) and feature map are both one-dimensional. As an example, for an input with the size of 1×19, with a filter size of 4, number of filters of 4 and stride of 3, the output of the convolution layer reaches the size of 1×6×4 (i.e. for each filter, the input vector of size 1×19 is converted into 6 features, based on convolution of elements 1–4, 4–7, 7–10, 10–13, 13–16, 16–19) as can be observed in Figure 5. As seen, the 1×4 kernel undergoes scalar multiplication with every four numbers, outputting one number every time. The 1-D convolution extracts features as follows:[32]

$$x_k^{(i)} = \sum_c w_k^{(i),c} \times x^{(i-1),c} + b_k^{(i)} \tag{5}$$

At the i^{th} layer (i equal to 2 in Figure 5), k is the index for a specific feature map, c refers to the channel number of the input $x^{(i-1)}$. $w_k^{(i),c}$ is the k^{th} convolution kernel corresponding to the c^{th} channel, $b_k^{(i)}$ refers to the bias of the k^{th} feature map.

As suggested by Ioffe and Szegedy,[33] batch normalisation should be carried out before activation to get rid of the distribution shift. The BN layer applies a transformation that maintains the mean of the convolved features close to zero and the variance of the convolved features close to one. It normalises its inputs $x_k^{(i-1)}$ (the input at k^{th} feature map) via using the computed mean μ and variance σ^2 of a mini-batch (i.e., subset of the training set) and over each input channel as follows:

$$\widehat{x_k^{(i)}} = \frac{x_k^{(i-1)} - \mu}{\sqrt{\sigma^2 + \varepsilon}} \tag{6}$$

where ε is suggested in the case of a small mini-batch variance in order to improve numerical stability. In the situation that inputs with a mean of zero and variance of one are not suitable for the subsequent layer, the batch normalisation layer can be shifted and scaled as below:

$$y_k^{(i)} = \gamma \widehat{x_k^{(i)}} + \beta \tag{7}$$

Here, the offset β and scale factor γ are learnable parameters that are updated during network training. The normalised features are input into a layer with ReLU activation function $F(\cdot)$, calculated as below:

$$F(x) = \max(0, x) \tag{8}$$

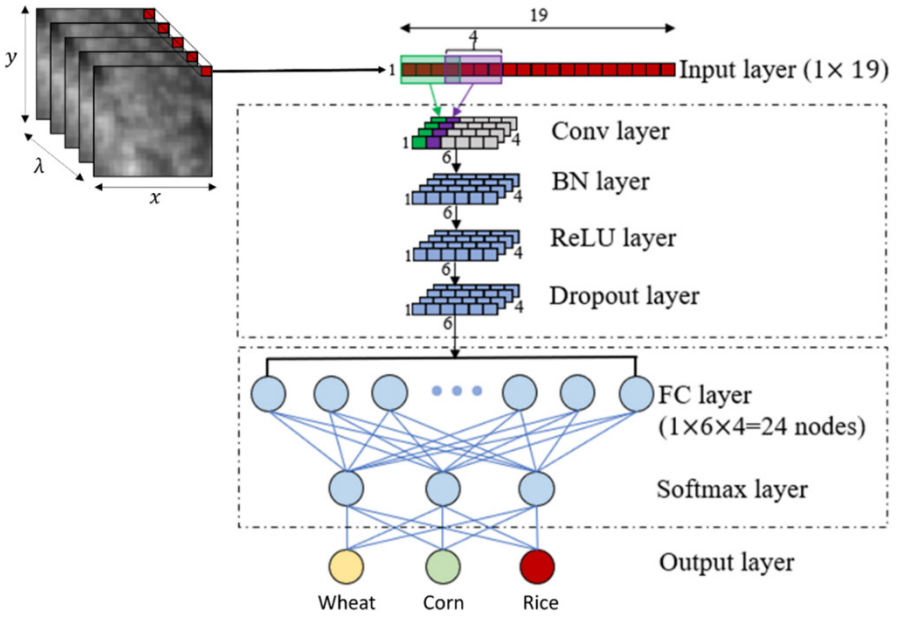

Figure 5. Architecture of the 1-dimensional CNN model consisting of an input layer (size of 1 ×19), a convolution (Conv, filter size=[1 4], number of filters = 4, stride = 3) layer, a batch normalisation (BN) layer, a rectified linear unit (ReLU) layer, a dropout layer, a fully connected (FC) layer, a softmax layer and an output layer.

This is followed by a dropout layer which is a regularisation method to prevent a model from overfitting. Srivastava *et al.*[34] proposed the strategy of dropping units, i.e., neurons over network training to reduce overfitting. The choices of dropout neurons are random with a given probability, defined by the user. After the dropout layer, fully connected (FC) layer is used to merge all feature maps (i.e., four feature maps from Figure 5). Therefore, the number of neural nodes depends on the convolution kernel size, the sampling kernel size and the number of feature maps. In this case, the number of nodes is $1 \times 6 \times 4 = 24$, as illustrated in Figure 5. In the FC layer, every neuron in the (i)-th layer is connected to every neuron in the subsequent layer ($I+1$)-th. For a multiclassification task, it is a common practice to place a softmax layer after the last FC layer. The input of softmax comes from K different neurons of the FC layer. Zhang *et al.*[30] has reported how to calculate the probability (P) that the independent variable x belongs to the j-th class as below:

$$P(y = j \mid x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}} \quad (9)$$

Selection of convolution parameters

First derivative (SG with a window size of 11 and third order polynomial degree) followed by SNV were applied to pixel spectra prior to 1-D CNN model development. Similar to PLS-DA modelling, pixel spectra were first extracted to form an $\mathbf{X}$ matrix, which together with the class membership $\mathbf{Y}$ matrix are then fed into the 1-D CNN architecture. In this work, a 2-D convolution layer is used with some modifications to perform 1-D convolution in MATLAB. The input for the 2-D convolution layer is an image with the size of height, width and the number of colour channels. To match the size of an input layer for a 1-D CNN, the $\mathbf{X}$ matrix with size of N (number of pixels, i.e. 14,589 pixels) $\times \lambda$ (spectral variables, i.e. 101 wavelengths) needs to be reshaped into the size of $1 \times \lambda \times 1 \times N$, with the height and the number of colour channels replaced as 1. Table 2 characterises some important training options during CNN model development. For the network training of the 1-D CNN model, the learning rate[35] is set to 0.01, the mini-batch size is set to 4096 and the number of max epochs of training is set to 100. Details of the entire set of training options can be found in Table S1.

Tuning parameters for DL requires expertise and extensive trial and error. In order to select suitable parameters for the 1-D CNN model developed here, we explore the influence of filter size, number of filters and stride on the performance of a 1-D CNN model, with the whole procedure recorded in "Section5_1D_CNN_model_parameters.m". The filter size is gradually

Table 2. Some important training options in DL model development.

Training option	Definition	Interpretation
Plot of training progress	The plot shows the mini-batch loss and accuracy against iteration.	Plot the progress of the network as it trains. This plot can be used to diagnose the occurrence of overfitting. For example, as the number of iterations increases, the training error gradually decreases, while the validation error decreases first and then increases, which implies the emergence of overfitting.
Max epochs	An epoch is the full pass of the training algorithm over the entire training set.	The more epochs specified, the longer the network will take to train, but the accuracy may improve with each epoch.
Mini-batch size	A mini-batch is a subset of the training data set that is processed at the same time in one iteration.	The larger the mini-batch, the faster the training, but the maximum size will be determined by the GPU memory. Reduce the mini-batch size if a memory error occurs.
Learning rate	A tuning parameter that is applied in an optimisation algorithm to decide the step size at each iteration while approaching a minimum of a loss function	This is a major parameter that controls the speed of training. A lower learning rate can give a more accurate result, but the network may take longer to train.

increased from 5 to 70 with an interval of 5, and the 1-D CNN models are developed keeping the other parameters constant (e.g., number of filters = 20, stride = 5). The classification accuracy for the validation set is plotted against filter size in Figure 6A. The model obtains the best classification result at the convolution kernel size of 10. As a result, the filter size is set to 10 for the subsequent tuning of parameters. The number of filters changes from 5 to 50 with at a step of 5 under the condition that other parameters are the same (e.g., filter size = 10, stride = 5). The accuracy of the validation set is plotted against the number of feature maps in Figure 6B. The accuracy raises rapidly at the beginning until it reaches 20, after which it tends to fluctuate. Some features that are significant to network learning are missing if the number of feature maps is too small, leading to poor classification performance.[30] However, inclusion of too many feature maps increases the model training time and risk of overfitting. As a compromise, 20 feature maps are selected. Another option is to optimise several parameters simultaneously based on the combination of parameters. As an example, there are 14 filter sizes ranging from 5 to 70 with the interval of 5 and 10 different numbers of filters changing from 5 to 50 at a step of 5, leading to 140 combinations (14×10) and, therefore, 140 models. The model performance in terms of accuracy of the validation set is shown in Figure 6C. The optimal stride, also known as sampling step size, is determined under the condition that the filter size is 10 and 20 filters are used for feature extraction. Generally, the size of stride is required to be smaller than the filter size. Hence, models are built at the increasing stride in steps of 1 from 1 to 10. As shown from Figure 6D, the best classification performance can be achieved when the stride is set to 4.

Classification performance

Finally, a 1-D CNN model can be developed based on the selected parameters above, i.e., filter size = 10, number of filters = 20, stride = 4, as can be found in the script entitled "Section5_1D_CNN_final_model_performance". Generally, DL attempts to learn the correct distribution of the data and is prone to overfit the data at some point in time. Over the training process, the training error will keep decreasing, yet the validation error might show a different trend, e.g. decrease at the beginning stage and then increase, suggesting the occurrence of overfitting. Hence, this work applies early termination for 1-D CNN training via setting the validation patience at 5, which means that the training stops when it reaches 5 times that the loss of validation set is not less than the previously smallest loss.

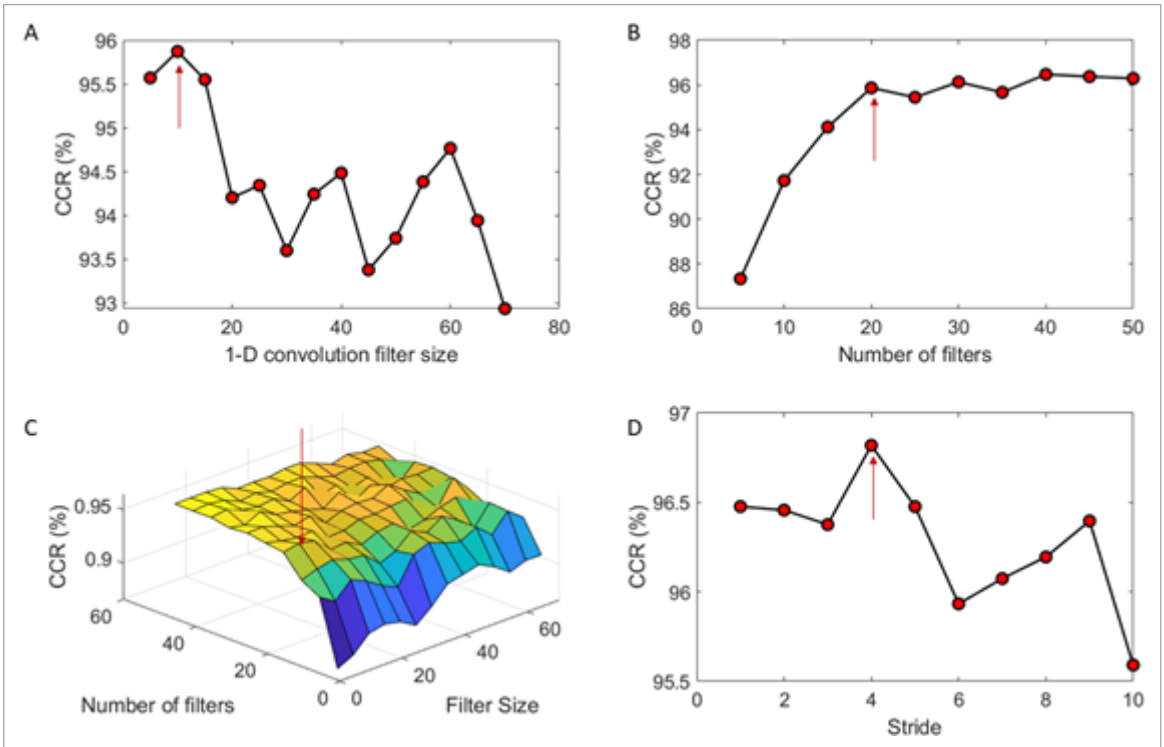

Figure 6. Classification performance of validation set at different filter sizes (A), number of filters (B), the combination of filter size and filter number (C) and strides (D) for 1-D CNN model. The selected parameter is indicated by the red arrow.

Figure 7 shows the accuracy and loss (i.e. the difference between the predicted and the real value) for training and validation sets plotted against the number of iterations. The accuracy for training and validation both increase and then remain flat. The evolution of a loss curve over the training process is usually used to diagnose the stability of a DL model. The loss of the model is generally lower on the training set than the validation set. A minimal gap between the two final loss values is preferred and identified as a good fit. As seen from Figure 7, the loss on the training set decreases rapidly for the first 20 iterations, suggesting that the network is learning fast to classify cereal samples. The loss of the validation set does not decrease as fast but stays roughly within the same range as the training loss, implying that this model generalises reasonably well to unseen data. It is also observed that high accuracy (CCR > 95 %) for training and validation are achieved with exceedingly reduced loss after 100 iterations.

Figure 8 shows confusion matrices for validation and test sets. Compared to Figure S3 which shows the results for the PLS-DA model applied to the same data (and pre-treatments), the 1-D CNN has improved classification performance in all aspects with higher CCR, TPR, PPV and lower FNR and FDR. The improvement is more significant for classification of corn and rice (Figure 8A), which agrees with the classification map (Figure 8B) where less pixels of rice have been wrongly identified as corn, compared to all PLS-DA models in Figure 3. In more detail, the 1-D CNN model incorrectly classified 412 pixels of rice as corn for the mixture image (Figure 8C), which is much less than that of PLS-DA models with the best one built from first derivative and SNV showing 710 pixels of rice wrongly classified as corn (Figure S4).

Three-dimensional CNN modelling

3-D CNN architecture

Spectral images are data-rich thanks to the integration of spatial and spectral information. However, most existing data analysis techniques tend to focus primarily or exclusively on the spectral domain,[36] and the spectral data is processed without considering the spatial features.[37] To overcome this shortcoming, the 3-D CNN has been proposed to extract high-level spectral–spatial features from the original 3D inputs. Pixel-based classification of a spectral image $I(x,y,\lambda)$ aims at predicting an individual pixel class. Since neighbouring pixels usually have the same labels, it is beneficial for the model to take the "spatial coherence" into account. In this sense, the first step of a 3-D CNN is to extract a $k \times k \times \lambda$ patch around each pixel, where k denotes the window size of the patch. Extraction of a patch could be performed using the

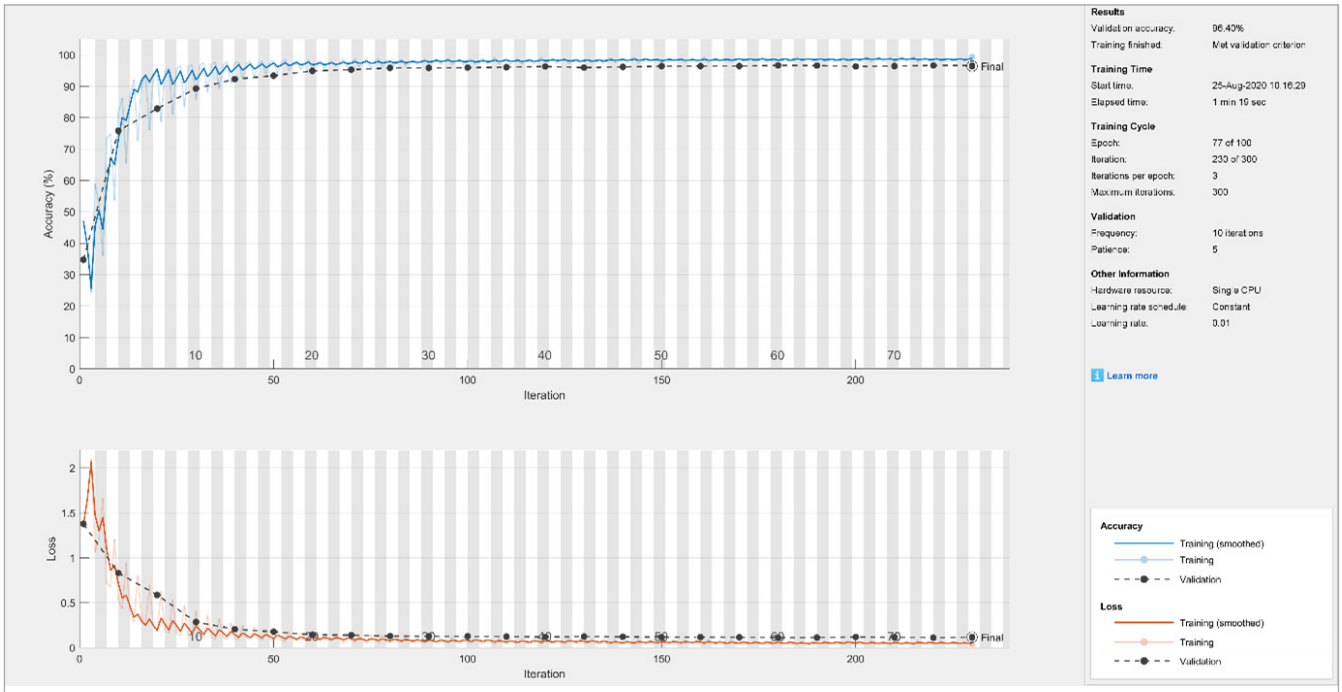

Figure 7. Training progress of 1-D CNN with accuracy (top) and loss (bottom) plotted against iteration. The number of epochs is indicated above the X axis.

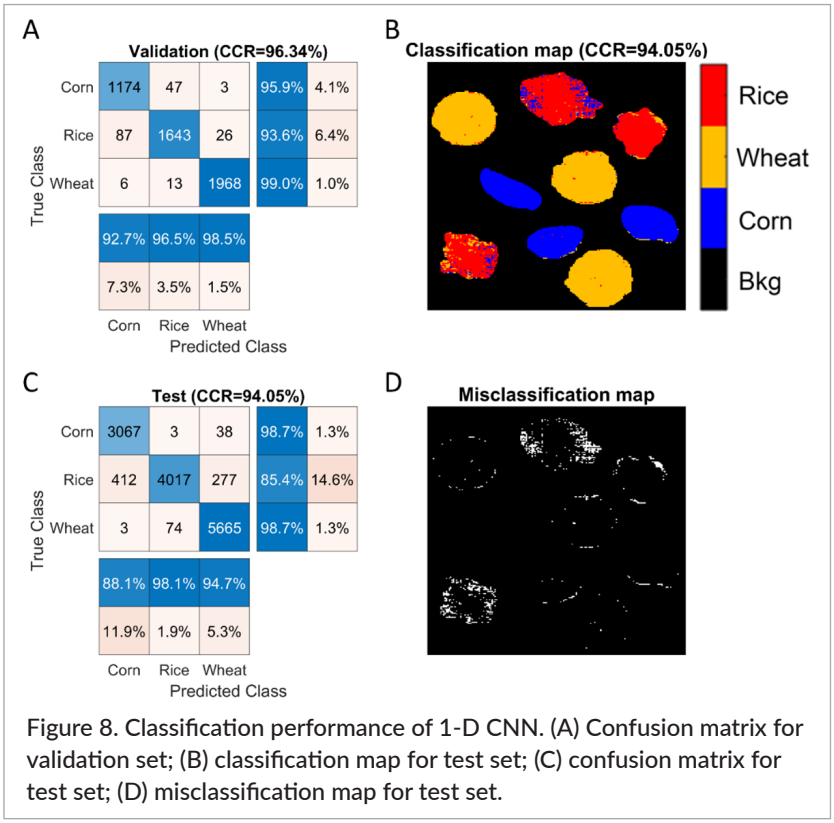

Figure 8. Classification performance of 1-D CNN. (A) Confusion matrix for validation set; (B) classification map for test set; (C) confusion matrix for test set; (D) misclassification map for test set.

original function "patch_extract_HSI". Specifically, each patch (i.e., the spatial/spectral context) is created by the neighbouring pixels surrounding a pixel, i.e. the centre point. The patch may include some pixels belonging to the sample while the others may belong to the image background when the pixels are distributed near the edge of the image. Therefore, background removal is carried out before patch extraction. Specifically, in the example provided, the background pixels are assigned to 0.

A schematic of a 3-D CNN architecture is illustrated in Figure 9. As an example, a patch with size of 7×7×101 is extracted from a spectral image and used as the input. As shown, it is quite similar to those of the 1-D CNN model, except that a 3-D convolution filter rather than

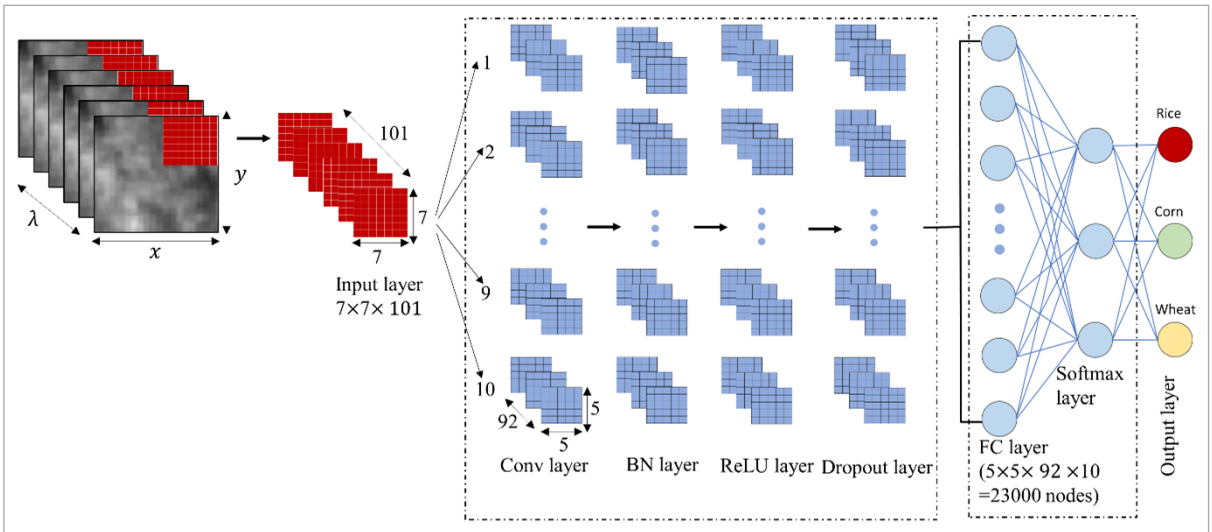

Figure 9. Architecture of the 3-dimensional CNN model consisting of an input layer (size of 7×7×101), a convolution (Conv, filter size = 3×3×10, number of filters = 10, stride = 1) layer, a batch normalisation (BN) layer, a rectified linear unit (ReLU) layer, a dropout layer, a fully connected (FC) layer, a softmax layer and an output layer.

1-D convolution operator is utilised. As seen from Figure 9, the filter size is set as $3 \times 3 \times 10$ and this 3D filter can move in all three directions (x,y,λ). Since the filter slides through a 3D space with a stride of 1, the output is arranged in a 3D space as well with a size of $5 \times 5 \times 92$. Similar to the 1-D CNN, a BN layer, ReLU layer, dropout layer, FC layer and softmax layer is included in the designed network structure.

Selection of patch window size

In this example, the same pre-treatments as used in the previous sections (i.e., first derivative pre-processing followed by SNV) were applied to the spectra prior to 3-D CNN model development. As illustrated schematically in Figure 9, an 8-layer 3-D CNN model was built and applied for the classification task. Just as for the 1-D CNN model, there are many parameters that need to be optimised. However, it should be noted that a considerable amount of training time is required to develop the 3-D CNN model and memory limitations might occur in the process (for example it takes 10 minutes for model development on a computer using Windows 10 Pro, processor: Intel®Core™i7-6700CPU@3.40 GHz; installed memory: 16.0 GB; 64-bit operating system). The spatial and spectral relationship with neighbouring pixels is the key to successful classification, due to pixels being classified based on features extracted from the surrounding patch. Therefore, it is worthwhile to assess the influence of patch window size on model performance. According to the previous section on selecting 1-D CNN parameters, the optimal filter size of convolution layer on the spectral domain should be 10, with 20 feature maps and moving at a stride of 4. As for the convolution filter size applied to the spatial dimensions, an odd-sized filter is usually used since all the previous layer pixels would be symmetrically arranged around the output pixel, which facilitates implementation simplicity. In general, smaller odd-sized kernel filters would be preferred, yet 1×1 is eliminated from the list of candidate optimal filter sizes as the features extracted would be fine grained and local, with no information from the neighbouring pixels. Therefore, in this example the filter size is set to $3 \times 3 \times 10$, the filter number is set to 20, and the stride is set to $1 \times 1 \times 4$. For the classification of each pixel, a $k \times k \times \lambda$ patch surrounding it is first constructed, where k changes from 3 to 15 in steps of 2, producing seven 3-D CNN models. The 3-D CNN input layer has the size of [h w d c], where h, w, d and c correspond to the height, width, depth and number of channels, respectively. Prior to model development, the size of the X matrix [originally $k \times k \times \lambda \times N$ (number of pixels)] is transformed into $k \times k \times \lambda \times 1 \times N$, to match the required size for the 3-D input layer, where 1 represents the number of channel. Classification accuracy is plotted against the window size of patch, as shown in Figure 10. It is observed that the maximum accuracy is achieved at window size of 5. The whole workflow is recorded in the script file "Section6_3D_CNN_model_parameter.m".

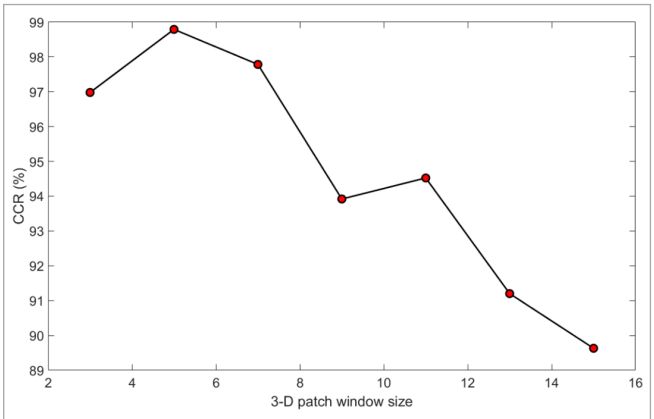

Figure 10. Classification accuracy performance of validation set at different patch window sizes for a 3-D CNN model.

Classification performance

Following patch size optimisation, a 3-D CNN model is subsequently developed (see the script of "Section6_3D_CNN_final_model_performance.m") based on the $5 \times 5 \times \lambda$ patch with the training progress shown in Figure 11. Compared to the 1-D CNN model (Figure 7), similar curve shapes (i.e., accuracy and loss) are noticeable, where accuracy first soars and then remains constant after 100 iterations, while loss declines rapidly at the beginning and then remains flat, indicating that the model is not under or over-fitted. A smaller gap of accuracy and loss between training and validation is evidenced for the 3-D CNN model, suggesting a better capability of model generalisation on unknown data.

Confusion matrices for validation and test sets were computed and are illustrated in Figure 12A and C, respectively. Impressive classification results are obtained compared with the previous approaches described. We can observe that a much higher accuracy is obtained, namely, accuracy of 99.09% for validation and 99.23% for the test set. It is a huge improvement from CCR of 91.56% using PLS-DA, 92.53% using Ctree and CCR of 94.05% using 1-D CNN for prediction of the mixture image. Apart from this, higher TPR, PPV and lower FNR and FDR can be seen for the 3-D CNN model compared to PLS-DA and the 1-D CNN model, confirming the

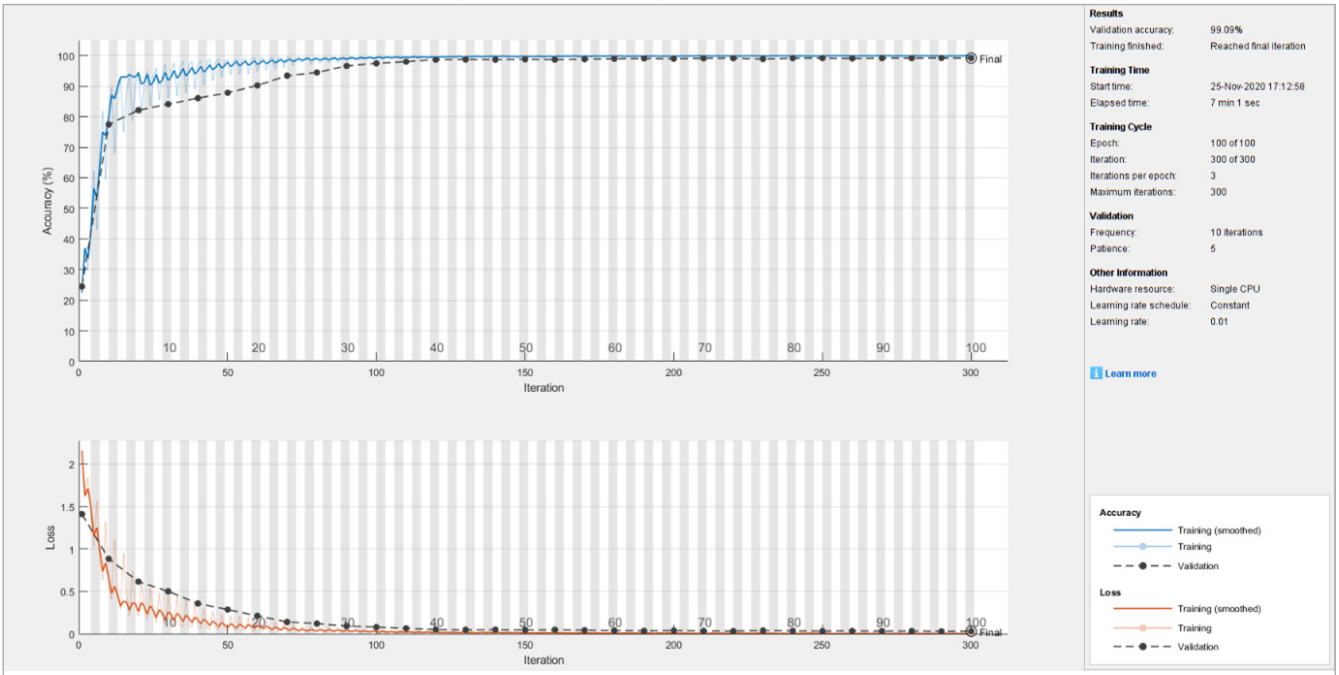

Figure 11. Training progress of 3-D CNN with accuracy (top) and loss (bottom) plotted against iteration.

superior performance of 3-D CNN in every aspect. As shown from the prediction maps (Figure 12B), the number of pixels that are correctly classified is extremely high; for instance, there are much fewer rice pixels incorrectly assigned to corn. The outstanding performance can be further supported by the misclassification map (Figure 12D) where only a few misclassified pixels can be seen.

The 3-D CNN model is generally difficult to interpret[17] due to the "black box" nature of the training procedure.

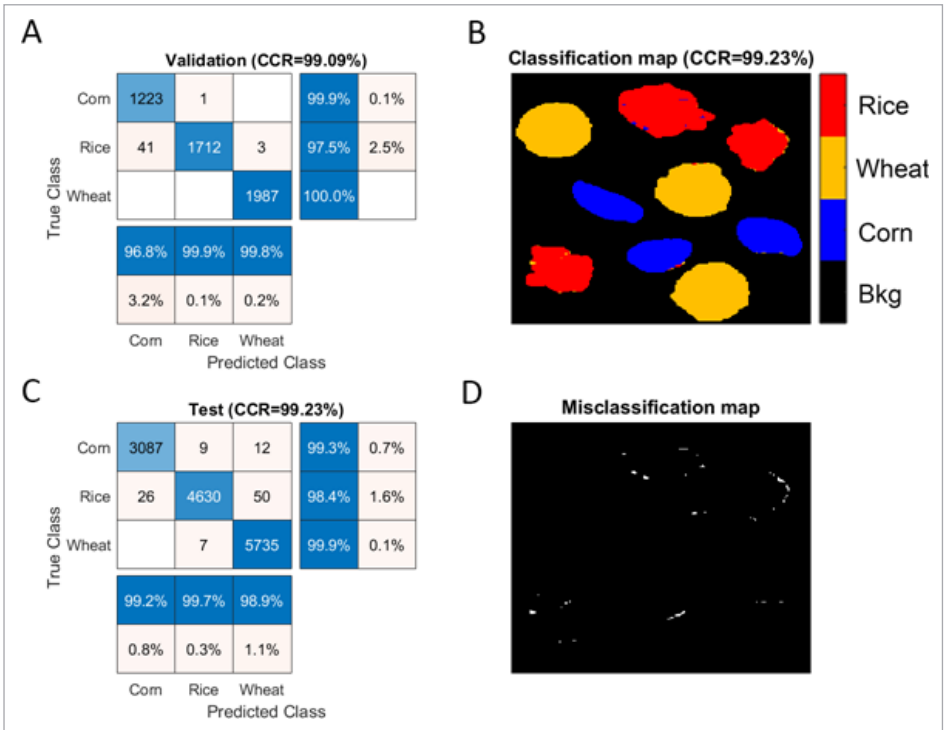

Figure 12. Classification performance of 3-D CNN. (A) Confusion matrix for validation set; (B) classification map for test set; (C) confusion matrix for test set; (D) misclassification map for test set.

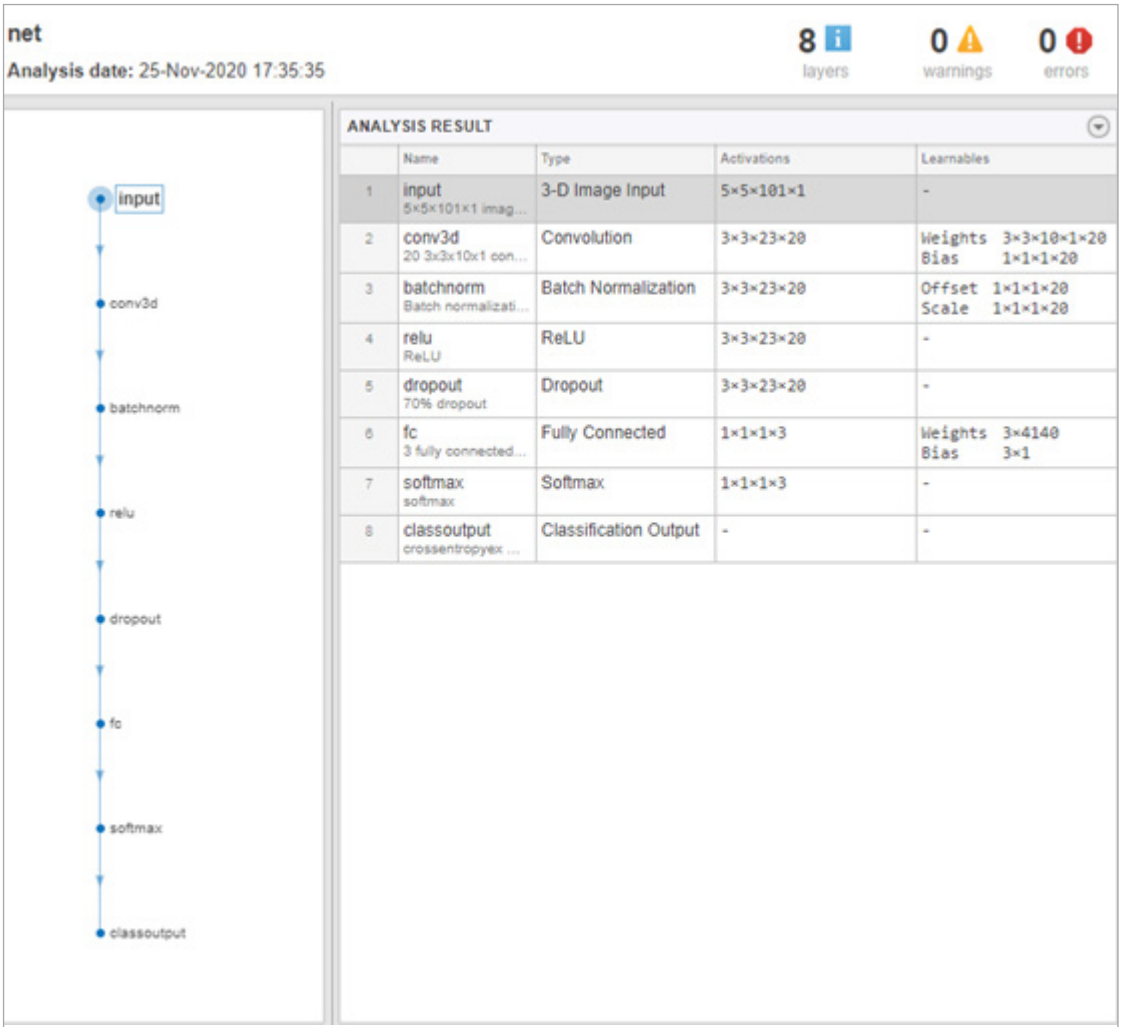

Figure 13. The output plot obtained after applying *analyzeNetwork* function on the 3-D CNN trained network.

Therefore, this tutorial also demonstrates some basic strategies to visualise and understand the architecture of a network, as well as the extracted features from different layers. Several efforts have been made to "open" the box and understand the design.[38] For instance, *analyzeNetwork* is a useful MATLAB function that facilitates visualisation of the neural network architecture and detection of errors and issues in the network. As an example, see Figure 13. The *analyzeNetwork* function displays an interactive plot of the network architecture (on the left) and a table containing information about the network layers (on the right) which includes the sizes of layer activations and learnable parameters.

In addition, this tutorial shows how to feed a $k \times k \times \lambda$ patch to the convolutional neural network and display the output activations of different layers of the network, in order to examine the activations and discover which feature the network has learned. Three pixels (labelled as P1, P2 and P3 in Figure 14) belonging to each class are selected from the training set, and the patch with the size of $5 \times 5 \times 101$ surrounding each pixel is subsequently extracted. Each patch is displayed in 3-D view (Figure 14), where the X and Y axes represent spatial dimensions (5×5) while Z refers to the spectral dimension with 101 wavelength variables. Spectra can be extracted from each patch and are plotted in Figure S5. It is found that the Rice-P1 patch contains some background pixels with all zero values over the entire spectral region as well as some abnormal spectral profiles. Each patch ($5 \times 5 \times 101$) is fed into the 3-D CNN network as the input. After each patch passes through the Conv layer, the output activations are returned as a 4-D array with the size of $3 \times 3 \times 23 \times 20$ (see Figure 13), with the fourth dimension indexing the number of filters. Given the fact that the original 5×5 image at each wavelength plane is transferred to 3×3 image after convolution processing, this

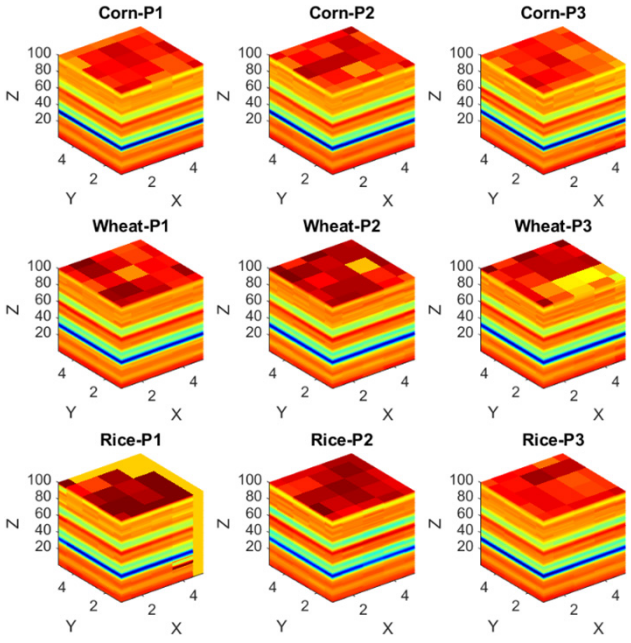

Figure 14. The 3-D view of individual patch with the size of 5 × 5 × 101.

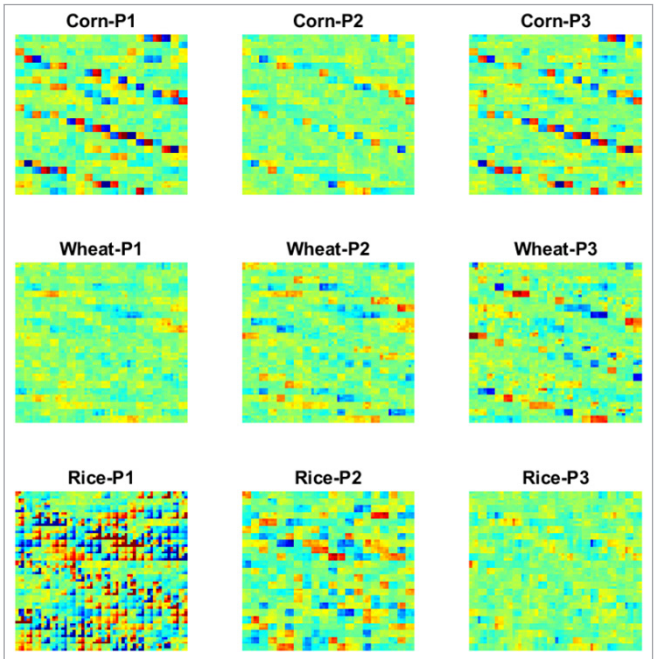

Figure 15. Visualisation of the output activations of the convolution layer.

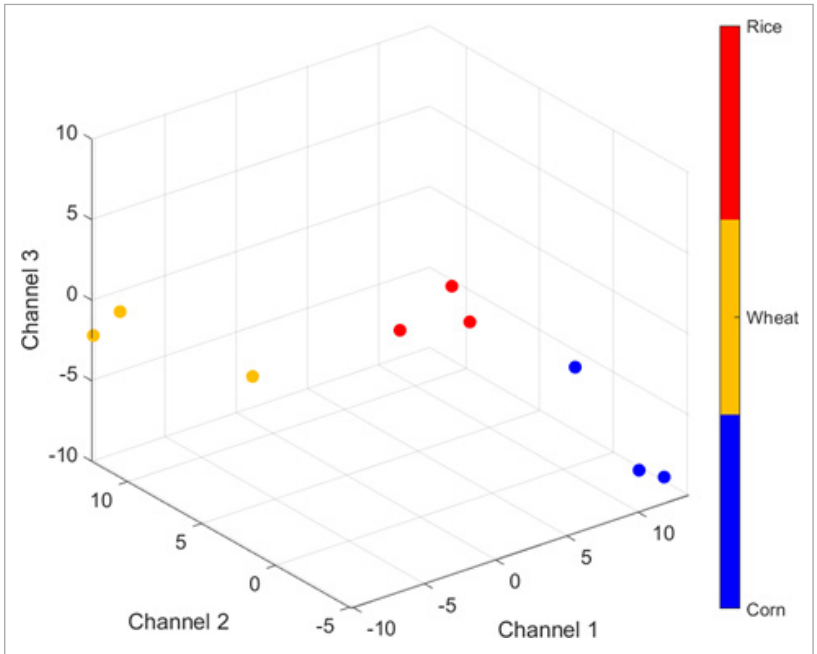

Figure 16. Visualisation of the output activations of the fully connected layer.

leads to 23 × 20 = 460 images in total. For the purpose of visualisation, these 460 images are displayed in a 23 × 20 grid in each subplot of Figure 15. Red pixels represent strong positive activations and blue pixels refer to strong negative activations. A red pixel at some location in a channel (i.e. feature map) indicates that the channel is strongly activated at that position. It can be observed that there is a similar pattern from three patches of Corn. In addition, Rice-P1 demonstrates unusual values probably because it comes from the edge region. Likewise, the output activations of the dropout layer are presented in Figure S6. The fully connected (FC) layer combines the features from the previous layer and aims to reach a classification decision. This tutorial extracts the output activations of the FC layer and plots them in Figure 16. The output argument of the FC layer of the network is

equal to the number of classes of the data set, therefore, each patch at this stage is represented by a single value at three channels indexing each class (see Figure 13). Good separation among classes is observed in Figure 16, implying that the representative features are strongly activated for individual class.

PLS-DA and Dtree modelling built from patch extraction

Focusing on the fact that the 3-D CNN model is built and applied on a 3-D patch with the size of $k \times k \times \lambda$, readers might presume that the extraordinary predictive ability originates from the use of a set of pixel spectra within the patch instead of the conventional way of using a solely pixel spectrum. For a fairer comparison, PLS-DA with six LVs selected (see Figure S7) and Dtree models are developed based on the mean spectrum of each patch with the same size of $5 \times 5 \times \lambda$ (see the script of "Section7_PLS-DA_Dtree_from_patch_images.m"), with results shown in Figures S8 and S9, respectively. The same pre-processing procedures, i.e., first derivative (SG with a window size of 11 and third order polynomial degree) and then SNV, are applied on spectra before model development. An improvement is evidenced for validation when compared to original PLS-DA models (see Figures S3 and S4). Nevertheless, the performance for the test set is slightly enhanced from CCR of 91.56 % (SNV and derivative processed on pixel spectra) to CCR of 92.33 %. When looking at the classification and misclassification maps (Figure S8B and D, respectively), misclassified pixels tend to form clusters and randomly distributed single misclassified pixels are largely reduced, because the approach of using mean spectra of the surrounding pixels, also known as local averaging, enables filtering of random noise. But, still a large number of rice pixels are wrongly classified as corn, suggesting the insufficiency of PLS-DA compared to 3-D CNN modelling. It is also evident that the Dtree classifier is superior to PLS-DA with accuracy of 96.74 % for the test set (see Figure S9).

Challenges of 3-D CNN modelling

PLS-DA, Dtree and 1-D CNN models are developed based on the traditional pixel-wise methods, which utilise only the spectral information from each pixel, while 3-D CNN extracts combined spectral–spatial features. The results from this work confirmed that joint spectral spatial features can be more effective than considering solely spectral features, therefore, the 3-D CNN model should be further investigated as a powerful classification tool in NIR spectral imaging. Despite the strong predictive ability, 3-D CNN model development requires a long computational time due to the complexity of deep neural network layers. Indeed, 3-D CNN modelling brings complexity into the classifier, requiring more parameters that need optimisation during model development. In the following, we summarise some of the major challenges for 3-D CNN models:

1) The training of 3-D CNNs could be a computational burden, since it usually requires computationally expensive and memory-intensive methods,[39] due to the extent of parameters that must be managed for massive spectral imaging datasets.
2) A 3-D CNN model trained from spectral imaging data might be ineffective in generalising the distribution of data, due to the fact that spectral imaging data is high-dimensional and often only a small sample size for training is available.[17]
3) The 3-D CNN model is more difficult to interpret[17] compared to PLS-DA. However, this tutorial demonstrates several useful strategies to understand the learned features from different layers.

Summary and suggestions

This tutorial provides a framework for development of 1-D CNN and 3-D CNN classifiers using NIR spectral imaging data. The main reason why the 3-D CNN model outperformed PLS-DA and the 1-D CNN is that it considers the joint spectral and spatial features. PLS-DA, Dtree and 1-D CNN architecture are usually used for spectral analysis on each pixel. Whereas the 3-D CNN model allows for the extraction and application of the combined spatial and spectral features from the spectral imaging dataset through the use of a 3-D patch and by applying 3-D convolution operators. Below we provide some suggestions regarding the general application of DL on spectral image datasets:

1) It is important to invest time in exploring the raw data (e.g., checking if there is any spatial information relevant for classification and identifying artefacts) and removing spectral and spatial backgrounds from the data, prior to modelling.

2) Assess possible pre-processing strategies and choose the most suited one(s) depending on the purpose. The use of standardisation pre-processing (e.g., SNV) might overcome some problems such as overfitting and the vanishing gradient when there is limited availability of training samples, or when very deep structures are designed.
3) Applying DL to high dimensional spectral imaging data is highly demanding in terms of computational workload. Therefore, advanced and powerful computers and hardware platforms should be used.
4) DL in general needs a large size of training data to fit an accurate and robust model. A lack of data often leads to overfitting due to the limited supply of examples that the network can learn from. If it is not possible to acquire more spectral imaging data, data augmentation techniques, which apply automatic transformations to images such as horizontal image flips, cropping, translations and rotation, can be performed to expand the amount of training data.
5) Pay attention to the problem of model overfitting. Generally speaking, predictive models should be validated on independent datasets. In addition, many techniques have been proposed to limit overfitting, including adding dropout layers (as is the case in this work), applying regularisation and reducing the network's capacity by removing some hidden layers.
6) Reduce the dimension of the massive spectral dataset prior to CNN modelling. The high dimensional nature of spectral images can result in difficulties in data storage as well as data processing. Dimensional reduction in spectral imaging without losing significant information about objects could ease the workloads during DL training. For example, principal component analysis (PCA) can be applied on spectral imaging data to reduce the data dimension. Afterwards, CNN modelling can be performed on the first several principal components instead of the entire dataset.

Acknowledgements

Funding for this research was provided by the European Research Council (ERC) under the starting grant programme ERC-2013-StG call-Proposal No. 335508-BioWater; and Science Foundation Ireland (SFI) under the investigators programme Proposal ID 15/IA/2984-HyperMicroMacro.

References

1. M. Brell, K. Segl, L. Guanter and B. Bookhagen, "3D hyperspectral point cloud generation: Fusing airborne laser scanning and hyperspectral imaging sensors for improved object-based information extraction", *ISPRS J. Photogram. Remote Sens.* **149,** 200–214 (2019). https://doi.org/10.1016/j.isprsjprs.2019.01.022
2. J. Abdulridha, O. Batuman and Y. Ampatzidis, "UAV-based remote sensing technique to detect citrus canker disease utilizing hyperspectral imaging and machine learning", *Remote Sens.* **11(11),** 1373 (2019). https://doi.org/10.3390/rs11111373
3. I. Orrillo, J.P. Cruz-Tirado, A. Cardenas, M. Oruna, A. Carnero, D.F. Barbin and R. Siche, "Hyperspectral imaging as a powerful tool for identification of papaya seeds in black pepper", *Food Control* **101,** 45–52 (2019). https://doi.org/10.1016/j.foodcont.2019.02.036
4. R. Rojas-Moraleda, N.A. Valous, A. Gowen, C. Esquerre, S. Härtel, L. Salinas and C. O'Donnell, "A frame-based ANN for classification of hyperspectral images: assessment of mechanical damage in mushrooms", *Neural Comput. Appl.* **28(1),** 969–981 (2017). https://doi.org/10.1007/s00521-016-2376-7
5. G.L. Alexandrino, J.M. Amigo, M.R. Khorasani, J. Rantanen, A.V. Friderichsen and R.J. Poppi, "Unveiling multiple solid-state transitions in pharmaceutical solid dosage forms using multi-series hyperspectral imaging and different curve resolution approaches", *Chemometr. Intell. Lab. Syst.* **161,** 136–146 (2017). https://doi.org/10.1016/j.chemolab.2016.11.004
6. L.M. Kandpal, J. Tewari, N. Gopinathan, P. Boulas and B.-K. Cho, "In-process control assay of pharmaceutical microtablets using hyperspectral imaging coupled with multivariate analysis", *Anal. Chem.* **88(22),** 11055–11061 (2016). https://doi.org/10.1021/acs.analchem.6b02969
7. C. Malegori, E. Alladio, P. Oliveria, C. Manis, M. Vincenti, P. Garofano, F. Barni and A. Berti, "Identification of invisible biological traces in forensic evidences by hyperspectral NIR imaging combined with chemometrics", *Talanta* **215,** 120911 (2020). https://doi.org/10.1016/j.talanta.2020.120911
8. A. Polak, T. Kelman, P. Murray, S. Marshall, D.J.M. Stothard. N. Eastaugh and F. Eastaugh, "Hyperspectral imaging combined with data classification techniques as an aid for artwork

authentication", *J. Cult. Herit.* **26,** 1–11 (2017). https://doi.org/10.1016/j.culher.2017.01.013

9. F. Rosi, C. Miliani, R. Braun, R. Harig, D. Sali, B.G. Brunetti and A. Sgamellotti, "Noninvasive analysis of paintings by mid-infrared hyperspectral imaging", *Angew. Chem. Int. Edit.* **52(20),** 5258–5261 (2013). https://doi.org/10.1002/anie.201209929
10. Y. Wang, X. Hu, Z. Hou, J. Ning and Z. Zhang, "Discrimination of nitrogen fertilizer levels of tea plant (*Camellia sinensis*) based on hyperspectral imaging", *J. Sci. Food Agric.* **98(12),** 4659–4664 (2018). https://doi.org/10.1002/jsfa.8996
11. S. Song, D. Gibson, S. Ahmadzadeh, H.O. Chu, B. Warden, R. Overend, F. Macfarlane, P. Murray, S. Marshall, M. Aitkenhead, D. Bienkowski and R. Allison, "Low cost hyper-spectral imaging system using linear variable bandpass filter for agriculture applications", *Appl. Optics* **59(5),** A167–A175 (2020). https://doi.org/10.1364/AO.378269
12. G. Elmasry, M. Kamruzzaman, D.-W. Sun and P. Allen, "Principles and applications of hyperspectral imaging in quality evaluation of agro-food products: a review", *Crit. Rev. Food Sci. Nutrit.* **52(11),** 999–1023 (2012). https://doi.org/10.1080/10408398.2010.543495
13. M. Barker and W. Rayens, "Partial least squares for discrimination", *J. Chemometr.* **17(3),** 166–173 (2003). https://doi.org/10.1002/cem.785
14. H. Wold, "Soft modelling by latent variables: the non-linear iterative partial least squares (NIPALS) approach", *J. Appl. Probab.* **12(S1),** 117–142 (1975). https://doi.org/10.1017/S0021900200047604
15. C. Garrido-Novell, D. Pérez-Marin, J.M. Amigo, J. Fernández-Novales, J.E. Guerrero and A. Garrido-Varo, "Grading and color evolution of apples using RGB and hyperspectral imaging vision cameras", *J. Food Eng.* **113(2),** 281–288 (2012). https://doi.org/10.1016/j.jfoodeng.2012.05.038
16. E. Ivorra, J. Girón, A.J. Sánchez, S. Verdú, J.M. Barat and R. Grau, "Detection of expired vacuum-packed smoked salmon based on PLS-DA method using hyperspectral images", *J. Food Eng.* **117(3),** 342–349 (2013). https://doi.org/10.1016/j.jfoodeng.2013.02.022
17. M. Paoletti, J.M. Haut, J. Plaza and A. Plaza, "Deep learning classifiers for hyperspectral imaging: A review", *ISPRS J. Photogram. Remote Sens.* **158,** 279–317 (2019). https://doi.org/10.1016/j.isprsjprs.2019.09.006
18. J. Schmidhuber, "Deep learning in neural networks: an overview", *Neural Networks* **61,** 85–117 (2015). https://doi.org/10.1016/j.neunet.2014.09.003
19. X.X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu and F. Fraundorfer, "Deep learning in remote sensing: a comprehensive review and list of resources", *IEEE Geosci. Remote Sens.* **5(4),** 8–36 (2017). https://doi.org/10.1109/MGRS.2017.2762307
20. A.J. Guo and F. Zhu, "A CNN-based spatial feature fusion algorithm for hyperspectral imagery classification", *IEEE Trans. Geosci. Remote Sens.* **57(9),** 7170–7181 (2019). https://doi.org/10.1109/TGRS.2019.2911993
21. A.A. Gowen, J.-L. Xu and A. Herrero-Langreo, "Comparison of spectral selection methods in the development of classification models from visible near infrared hyperspectral imaging data", *J. Spectral Imaging* **8,** a4 (2019). https://doi.org/10.1255/jsi.2019.a4
22. J.-L. Xu and D.-W. Sun, "Computer vision detection of salmon muscle gaping using convolutional neural network features", *Food Anal. Meth.* **11(1),** 34–47 (2018). https://doi.org/10.1007/s12161-017-0957-4
23. R. Barnes, M.S. Dhanoa and S.J. Lister, "Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra", *Appl. Spectrosc.* **43(5),** 772–777 (1989). https://doi.org/10.1366/0003702894202201
24. P.A. Gorry, "General least-squares smoothing and differentiation of nonuniformly spaced data by the convolution method", *Anal. Chem.* **63(5),** 534–536 (1991). https://doi.org/10.1021/ac00005a031
25. J.-H. Cheng and D.-W. Sun, "Rapid and non-invasive detection of fish microbial spoilage by visible and near infrared hyperspectral imaging and multivariate analysis", *LWT-Food Sci. Technol.* **62(2),** 1060–1068 (2015). https://doi.org/10.1016/j.lwt.2015.01.021
26. J.-S. Cho, H.-J. Bae, B.-K. Cho and K.-D. Moon, "Qualitative properties of roasting defect beans and development of its classification methods by hyperspectral imaging technology", *Food Chem.* **220,** 505–509 (2017). https://doi.org/10.1016/j.foodchem.2016.09.189
27. R.G. Brereton and G.R. Lloyd, "Partial least squares discriminant analysis for chemometrics and metabolomics: how scores, loadings, and weights differ according to two common algorithms", *J. Chemometr.* **32(4),** e3028 (2018). https://doi.org/10.1002/cem.3028

28. J.M. Amigo, H. Babamoradi and S. Elcoroaristizabal, "Hyperspectral image analysis. A tutorial", *Anal. Chim. Acta* **896,** 34–51 (2015). https://doi.org/10.1016/j.aca.2015.09.030

29. A.A. Gowen, G. Downey, C. Esquerre and C.P. O'Donnell, "Preventing over-fitting in PLS calibration models of near-infrared (NIR) spectroscopy data using regression coefficients", *J. Chemometr.* **25(7),** 375–381 (2011). https://doi.org/10.1002/cem.1349

30. L. Zhang, X. Ding and R. Hou, "Classification modeling method for near-infrared spectroscopy of tobacco based on multimodal convolution neural networks", *J. Anal. Methods Chem.* **2020,** 9652470 (2020). https://doi.org/10.1155/2020/9652470

31. J.L. Chu and A. Krzyżak, "Analysis of feature maps selection in supervised learning using convolutional neural networks", in *Advances in Artificial Intelligence. Canadian AI 2014*, Ed by M. Sokolova and P. van Beek. Lecture Notes in Computer Science, Vol. 8436, Springer (2014). https://doi.org/10.1007/978-3-319-06483-3_6

32. S. Nainan and V. Kulkarni, "Enhancement in speaker recognition for optimized speech features using GMM, SVM and 1-D CNN", *Int. J. Speech Technol.* (2020). https://doi.org/10.1007/s10772-020-09771-2

33. S. Ioffe and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv preprint 1502.03167 (2015). https://arxiv.org/abs/1502.03167

34. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", *J. Mach. Learn. Res.* **15(1),** 1929–1958 (2014).

35. D. Huang and M. Feng, "Understanding deep convolutional networks for biomedical imaging: a practical tutorial", *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Berlin, Germany, pp. 857–863 (2019). https://doi.org/10.1109/EMBC.2019.8857529

36. J. Burger and A. Gowen, "Data handling in hyperspectral image analysis", *Chemometr. Intell. Lab. Syst.* **108(1),** 13–22 (2011). https://doi.org/10.1016/j.chemolab.2011.04.001

37. A. Plaza, P. Martinez, J. Plaza and R. Perez, "Spatial/spectral analysis of hyperspectral image data", *IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data, 2003*. Greenbelt, MD, USA, pp. 298–307 (2003). https://doi.org/10.1109/WARSD.2003.1295208

38. P.E. Rauber, S.G. Fadel, A.X. Falcão and A.C. Telea, "Visualizing the hidden activity of artificial neural networks", *IEEE Trans. Visualiz. Comput. Graphics* **23(1),** 101–110 (2017). https://doi.org/10.1109/TVCG.2016.2598838

39. Y. Cheng, D. Wang, P. Zhou and T. Zhang, *A Survey of Model Compression and Acceleration for Deep Neural Networks*. arXiv preprint 1710.09282 (2017). https://arxiv.org/abs/1710.09282