Using deep learning to evaluate peaks in chromatographic data

Risum, Anne Bech; Bro, Rasmus

Published in:
Talanta

DOI:
10.1016/j.talanta.2019.05.053

Publication date:
2019

Document version
Publisher's PDF, also known as Version of record

Document license:
CC BY-NC-ND

Citation for published version (APA):
Risum, A. B., & Bro, R. (2019). Using deep learning to evaluate peaks in chromatographic data. *Talanta*, *204*, 255-260. https://doi.org/10.1016/j.talanta.2019.05.053

Talanta 204 (2019) 255–260

Contents lists available at ScienceDirect

Talanta

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

Using deep learning to evaluate peaks in chromatographic data

Anne Bech Risum, Rasmus Bro[*]

Department of Food Science, University of Copenhagen, Denmark

ARTICLE INFO

Keywords:
Deep learning
PARAFAC2
Expert system
Automation

ABSTRACT

Analysis of untargeted gas-chromatographic data is time consuming. With the earlier introduction of the PARAFAC2 (PARAllel FACtor analysis 2) based PARADISe (PARAFAC2 based Deconvolution and Identification System) approach in 2017, this task was made considerably more time-efficient. However, there are still a number of manual steps in the analysis which require data analytical expertise. One of these is the need to define whether or not each PARAFAC2 resolved component represents a peak suitable for integration. As the peaks may change in both shape and location on the elution time-axis, this presents a problem which cannot be readily solved by applying a linear classifier, such as PLS-DA (Partial Least Squares regression for Discriminant Analysis).

As part of our ongoing efforts to further automate analysis of Gas Chromatography with Mass Spectrometry (GC-MS), we therefore explore a convolutional neural network classifier, capable of handling these shifts and variations in shape. The theory of convolutional neural networks and application on vector samples is briefly explained, and the performance is tested against a PLS-DA classifier, a shallow artificial neural network and a locally weighted regression model.

The models are built on a training set with PARAFAC2 resolved components from eight different aroma related GC-MS runs with a total of over 70,000 elution profile samples, and validated using another, independent, GC-MS dataset.

Based on Receiver Operating Characteristic curves (ROC) and manual analysis of the misclassified cases, it is shown that the convolutional network consistently outperforms the competing models, yielding an Area Under the Curve (AUC) value of 0.95 for peak classification. Examples are given illustrating that this new approach provides convincing means to automatically assess and evaluate modelled elution profiles of chromatographic data and thereby remove this laborious manual step.

1. Introduction

Untargeted GC-MS can provide highly complex data. No specific chemical analytes are in focus a priori and hence it is not possible to optimize the chromatographic procedure as in classical analytical chemistry. It has been shown that the PARADISe approach [1], based on PARAFAC2, is capable of separating co-eluting peaks and baselines, outperforming state of the art analytical chemical software both in the number of quantified compounds, and in the reproducibility of the analysis. However, analysing GC-MS datasets using PARADISe still involves several manual steps, which are labour intensive and require a skilled tensor data analyst. The chromatograms are split into small intervals and a suitable PARAFAC2 model is fitted to the data from each interval. Some of the components of this model will represent baseline and others e.g. peaks of chemicals. Only the components representing chemicals are of interest. One of the most time-consuming tasks in PARADISe is the manual classification of PARAFAC2 components as being chemical peaks versus non-interesting phenomena such as baselines (see Fig. 1).

As part of our ongoing efforts towards automation and ease-of-use of the processing of GC-MS data, the purpose of the presented work was to develop an automated classifier, removing this laborious step in the workflow. At first glance, classifying the PARAFAC2 elution mode profiles might seem trivial. However, inherent problems in the nature of the data render our preferred chemometric methods unsuitable. The most significant issue is that the location and width of peaks on the time axis will shift considerably from different intervals. This type of nonlinearity cannot be handled well using PLS-DA, SIMCA, or other linear methods.

As an alternative, we therefore explore deep learning in the form of convolutional neural networks. The convolutional neural net searches for the same features in windows across the entire time axis, hence

[*] Corresponding author.
E-mail address: rb@food.ku.dk (R. Bro).

https://doi.org/10.1016/j.talanta.2019.05.053
Received 19 March 2019; Received in revised form 8 May 2019; Accepted 10 May 2019
Available online 22 May 2019

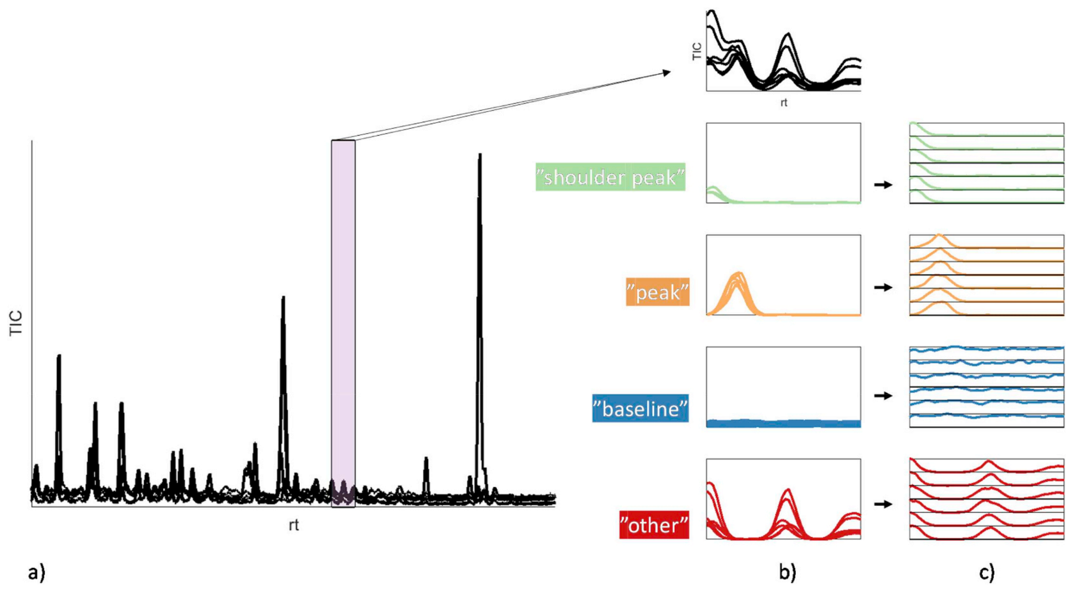

Fig. 1. Illustration of how the classified vectors are obtained from the GC-data. a) Example of a Total Ion Chromatogram of raw GC-MS data. The shaded area shows an example of an interval selected for PARAFAC2 modelling. Intervals are defined for each peak area in the chromatogram. b) Top: zoom on the selected interval below: elution profile loadings of the four PARAFAC2 components fitted on this interval, colored according to their classification. c) Illustration of how each of the sample elution profile loadings from the PARAFAC2 components become a single sample vector in the training data for the deep net.

allowing peaks similar in shape to shift but still be considered the same class; making it highly applicable for our problem. Furthermore, as large amounts of chromatographic data can easily be acquired, the need for large amounts of data in deep learning is not an issue in this case.

Here, we present a convolutional neural net approach capable of classifying the PARAFAC2 elution mode profiles into four classes: “peaks”, “baselines”, “shoulder peaks” and “other”, as illustrated in Fig. 1.

To evaluate the performance of the classifier, it is compared to PLS-DA as an example of a linear classifier, locally weighted regression (LWR) as a highly flexible non-linear model, and shallow artificial neural networks (ANN) as an alternative non-linear classifier.

2. Theory

The basic principles of deep learning in the form of convolutional neural networks will be shortly explained in the following. Normally, deep learning is applied to images and the theory is therefore usually described in terms of images. However, our input data will not be images with an x- and a y-plane. Instead our input will be elution profiles, hence vectors. Therefore, we will describe the theory in terms of such vectors. The basic principle of convolutional networks is that they make intensive use of a moving filter. The network consists of a number of layers starting with the input layer. The fact that there are many layers is what gives rise to the name *deep* learning. The input layer is where the data enters the network. Hence, if each sample is, say, a vector with fifty elements, then the input layer has fifty nodes – one node for each input variable.

The second layer of the network is then defined by a filter, for example of length two. The weights of this filter will be optimized during the training but for the example, assume that this filter reads $\mathbf{w} = [w_1\ w_1] = [0.5\ 0.5]$. For an input vector of length fifty that reads $\mathbf{x} = [7\ 3\ 5\ \ldots\ 3\ 9]$ the convolution/filtering can be depicted as in Fig. 2. The filtering runs from element (1,2), (2,3), (3,4) etc. of the input vector and takes the weighted sum. Hence, the first element in the following layer will be $7{\cdot}0.5 + 3{\cdot}0.5 = 5$. The fact that the same filter is applied only on a small segment at a time provides for some sparsity in the network which is important for the stability of the solution. That is, for a traditional feed-forward neural network each hidden layer is connected to

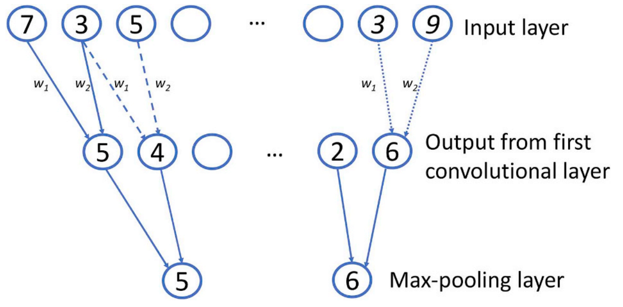

Fig. 2. An input layer followed by a convolution layer which is followed by a max-pooling layer. Normally there would also be first a batch normalization and then a ReLU layer before the max-pooling layer, but this is left out for simplicity.

the previous layer with a fully connected set of weights. If there are e.g. 100 variables in the previous layer, then there will be 100 weights to estimate. In the convolution layer above, there will be just two parameters regardless of the number of variables in the previous layer.

As opposed to a conventional fully connected neural network, the convolutional network looks for similar features defined by the weights in different parts of the input. The rationale for this has traditionally come from images where it is desired to be able to identify an object such as a chair no matter where in the picture it appears. The length of the filter, two in this case, is a parameter that needs to be established during training. It is usually much closer to two than to, say, ten. Hence, it is kept fairly small which may seem a little counter-intuitive especially, if the goal is, indeed, to identify chairs. However, there are many layers in the network and each layer is assumed to only handle very specific tasks. It is the aggregation over many layers of simple tasks that will allow the network to handle complex problems.

There are a number of hyperparameters (known as metaparameters in chemometrics) related to a convolutional layer. The filter width has already been mentioned but also the stride has to be defined. The stride is the step size which defines how much the filter moves. For example, if the stride is three, then the filter will start with element one and afterwards element four. Stride is often maintained at the value one as in Fig. 2. It is also possible to zero pad the input layer by adding zeros

Table 1
Overview of the size and origins of the different datasets used for training, stop set and validation of the deep neural network.

	GC-MS datasets	Intervals	PF2 components	Elution profile samples	Samples after preprocessing
Training data	1–8	618	1812	71078	119104
Stop set data	1–8	206	609	24796	43241
Validation set	9	381	1585	37961	37961[a]

[a] The Validation set data was kept as originally collected, only normalized, and not flipped or shifted in any way.

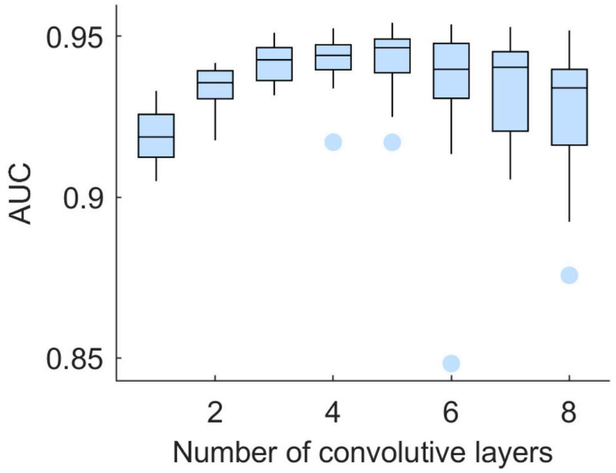

Fig. 3. Boxplot of the “peak” versus “non peak” classification AUC value for the deep convolutional nets as a function of the number of convolutive layers. For each level of number of convolutive layers, the model training was repeated 20 times.

before the first element and after the last.

After the convolutional layer, the data are normalized using batch normalization [2]. It is essentially an autoscaling and just a way to make sure that the output is within bounds. This helps speed up the algorithm. The normalization is done for each mini-batch. A mini-batch is a set of samples that are processed before updating of the parameters through backpropagation. The mini-batch size is often around some hundreds. After all samples have been processed once (termed one epoch), the data are usually shuffled before creating the next set of mini-batches. The shuffling tends to help in averaging out possible unfortunate distributions within the mini-batches.

The outputs from the batch normalization layer are normally passed through what is called a Rectified Linear Unit Activation function [3] referred to as a ReLU function. The ReLU function is defined $f(x) = \max(0,x)$. This simple cutoff helps the neural network model interactions as the output is only active above a certain threshold and also provides means for including certain non-linearities. The ReLU also has a bias implicitly added to x because the preceding convolutive layer has a bias. Hence, the actual cutoff that zero represents can be modified.

In a real neural network, there is not just one filter for each layer. Instead, there are several of these called channels. The number of channels is called the depth of the layer. By having many channels, it is possible to extract many different features rather than just one. This is somewhat comparable to having many components in a latent variable model.

After a filtering and a ReLU layer, it is common to have a so-called max-pooling layer. The aim of max-pooling is to do a re-sampling – typically a down-sampling. Imagine for simplicity that one channel has a feature that looks for ‘chairs’. Hence, any node with a high value will represent a chair. Max-pooling will use a filter where a set of neighbours are replaced with the max value. Hence, if just one of the ‘chair’-filters was high, then the node will have a high value (see Fig. 2). Apart from extracting potentially interesting information, the max-pooling will also filter out other information. Only the most significant features

Table 2
The structure of the final deep learning model. Convolutional neural network with four convolutional layers and 23 layers in total.

No.	Layer	Parameters
1	Image input layer	50 × 1 × 1 images, ‘zerocenter’ normalization
2	Convolution	20 channel, 3 × 1 convolutions with stride [1 1] and padding [1 1 0 0]
3	Batch normalization	
4	ReLU	
5	Max Pooling	3 × 1 max pooling with stride [1 1] and padding [0 0 0 0]
6	Convolution	40 channel, 3 × 1 convolutions with stride [1 1] and padding [1 1 0 0]
7	Batch normalization	
8	ReLU	
9	Max Pooling	3 × 1 max pooling with stride [1 1] and padding [0 0 0 0]
10	Convolution	60 channel, 3 × 1 convolutions with stride [1 1] and padding [1 1 0 0]
11	Batch normalization	
12	ReLU	
13	Max Pooling	3 × 1 max pooling with stride [1 1] and padding [0 0 0 0]
14	Convolution	80 channel, 3 × 1 convolutions with stride [1 1] and padding [1 1 0 0]
15	Batch normalization	
16	ReLU	
17	Max Pooling	3 × 1 max pooling with stride [1 1] and padding [0 0 0 0]
18	Fully connected	20 neurons fully connected layer
19	ReLU	
20	Dropout	25% dropout
21	Fully connected layer	4 neuron fully connected layer
22	Softmax	
23	Classification Output	Crossentropyex

are allowed through. Other functions than maximum can also be applied. Pooling is considered essential in convolutive networks for achieving the invariance to position and orientation of features within the data.

The output of the max-pooling layer is normally passed on to a new set of Conv/Normalize/ReLU/Max-pool layers. Often with less nodes to compress the information into the essential information but with more and more channels to be able to detect more complex phenomena. After the defined number of convolutive layers, a fully connected layer is added. This is similar to a normal feed forward neural network [4]. As in a normal neural network, it is possible to add several layers but typically relatively few layers are used. In deep learning it is common to add a drop-out layer after a hidden fully connected layer. The drop-out layer sets a fraction of the outputs to zero at random and provides a way to avoid overfitting, and forces the layer to be robust to missing out on certain features. The fraction to zero out is a hyperparameter that needs to be determined.

The final output layers will depend on the purpose of the network; typically, classification is the aim in which case the output layer can be a softmax layer which consist of a transfer function for each class using a normalized exponential which is a generalization of a logistic function [5]. It transfers a node value into a value between zero and one that can then be used for classifying.

Estimating the parameters of a network of a given structure can be

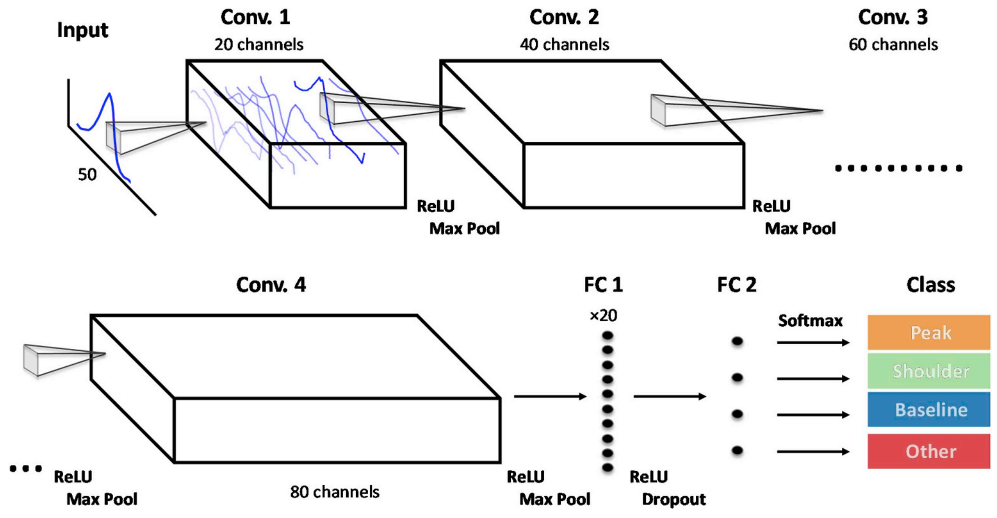

Fig. 4. Illustration of the architecture of the final deep network with four convolutional layers.

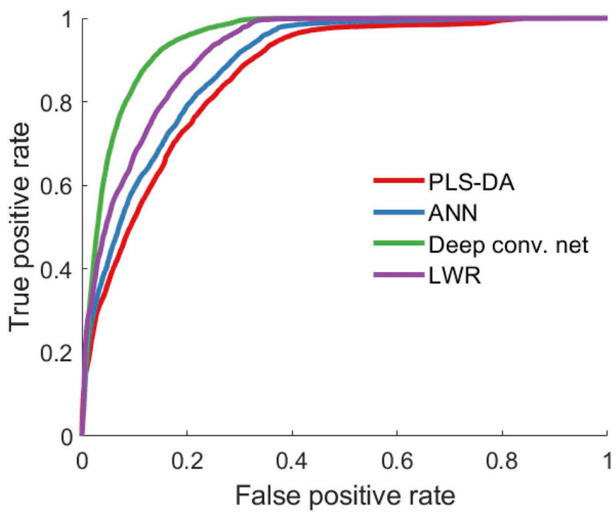

Fig. 5. ROC curves of the classification of “peak” versus “non peak” classes of the deep convolutional net compared to the other three tested models. The curves are based on the independent validation set.

done with classical back-propagation. Current state-of-the-art is an approach called adaptive moment estimation (ADAM) [6].

3. Materials and methods

3.1. Data collection

A total of nine different aroma related GC-MS datasets were analysed using PARADISe version 3.88 (http://www.models.life.ku.dk/paradise, March 12, 2019). The chromatograms were divided into intervals, each containing only a few peaks, which were then resolved by PARAFAC2 modelling. After choosing the appropriate number of components each component was manually classified into four classes: “peak”, “baseline”, “shoulder peak” and “other”. All elution profiles from the components were linearly interpolated to a length of 50 and collected as classified vectors used for input data in the deep convolutional net (see Fig. 1 for illustration of the workflow).

The dataset was divided into a training set, a stop set and a validationset. An overview of the dimensions of these datasets can be found in Table 1. Eight of the GC-MS datasets were used for the training data and stop set data. The stop set data was created by randomly selecting one fourth of the components from the training set,. The validation set was collected from a new independent GC-MS dataset.

3.2. Preprocessing

To create more variation in the training and stop set data, all vector samples were appended to the original data in a horizontally flipped version. This increased the dataset to double size. Furthermore, to introduce even more variation, all samples in the “other” class in the augmented matrix were shifted randomly with an integer from - − 10 to 10 points left or right on the x-axis, by moving the beginning of the vector to the end or the ending of the vector to the beginning, respectively. Similarly, all peak samples were shifted by adding a random integer of points (0–20) before and after the vector.

All sample sets, including the validation set, were re-sampled in order to have approximately 30% “peak” samples, 40% “other samples”, 15% “baseline” samples and 15% “shoulder peaks” samples. Finally, before any modelling, all of the vector samples in the appended, resampled matrices were normalized to length one. For the PLS-DA and LWR models, the data was also mean centred prior to modelling.

4. Deep learning

4.1. Selecting network structure

The AUC of peak versus non-peak classification as a function of number of convolutional layers is shown in Fig. 3. Simple networks with less than three layers did not perform well. Increasing up to three to five layers gave a substantial improvement whereas increasing complexity beyond that did not improve the average performance and resulted in less stable results.

The model settings described in Table 2 was chosen as a base model. An experimental design was run making slight modifications of step length, etc. It was found that the model was robust to slight modifications which we assume is because of the large amount of data used. Hence, no further optimization was pursued.

The final network, corresponding to the one outlined in Table 2, is illustrated in Fig. 4. It has four convolutional layers with 20, 40, 60 and 80 channels in each respective layer, from first to last. Stepsize is 3×1 throughout all convolutional layers, with a stride of one. Between each convolutional layer, a ReLU activation and MaxPooling (3×1) is performed. After the convolutional layers there is a fully connected layer with 20 neurons, followed by ReLU and a 25% dropout layer. Finally, a second fully connected layer with four neurons, a softmax activation and a classification layer yields the classification output.

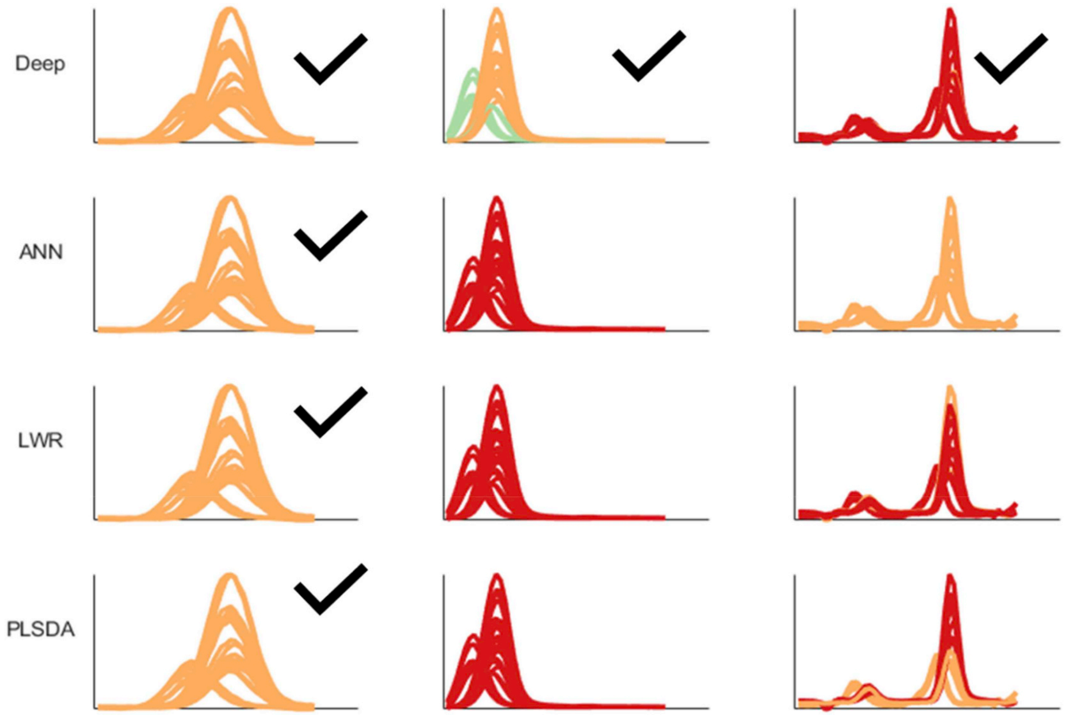

Fig. 6. Examples of classification results for the four methods. Each column shows one interval and each row one classification method. The classification result is indicated by the color (peak, shoulder peak, other). Correct classifications are indicated by tick marks.

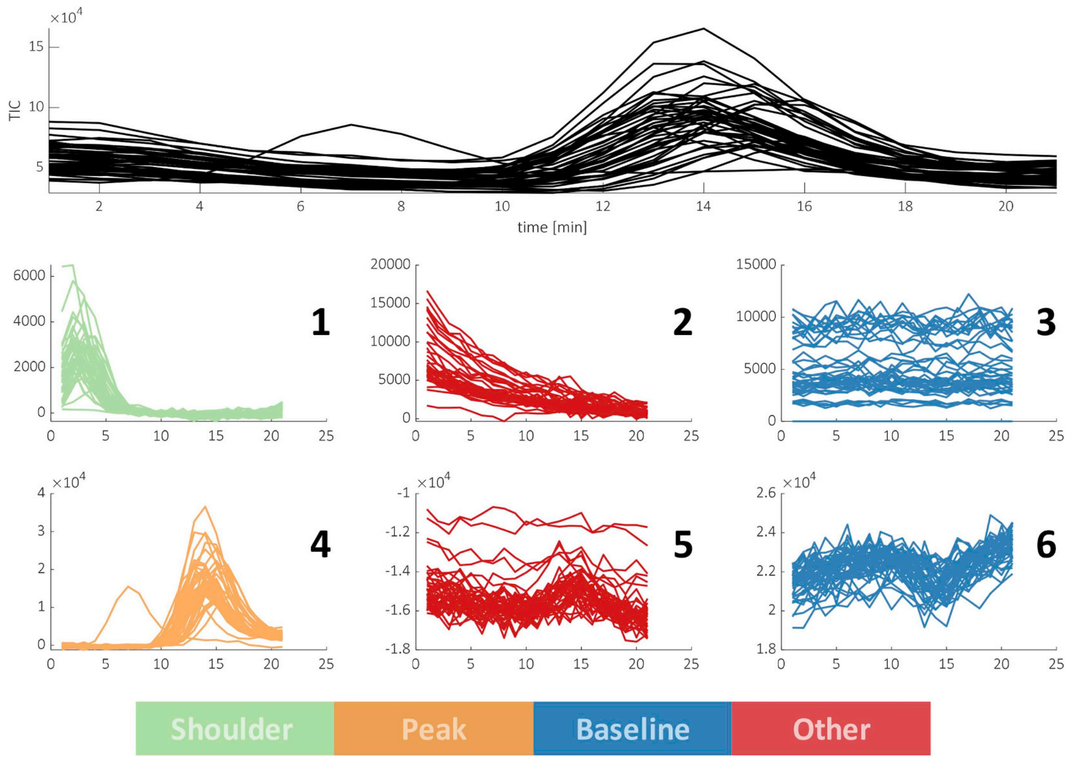

Fig. 7. Example of an interval fitted with a six component PARAFAC2 model, and subsequent deep net classification of all elution profiles in each component. Top: raw TIC. Bottom six: elution mode loadings of the six different components, colored according to the classification results. The numbers denote the component number.

4.2. Training the convolutional neural network

The network was trained using the neural network toolbox in Matlab 2018a. An ADAM optimizer was used to train. Initial learning rate was 0.1 and the learning rate drop period was ten with a learning rate drop factor of 0.1. The network was trained using minibatches with a minibatch size of 124 samples and shuffling between every epoch. The validation frequency was four times per epoch and validation patience was set at three. This means that the training stops when the validation using the stop set has not improved in three consecutive validation points.

5. Other algorithms for comparison

5.1. PLSDA

A PLSDA model classifying samples into the four classes was fit to the training data. Different number of components were tried, and the models were validated by prediction error on the stop set. The final model was fitted using three latent variables, and used to predict the validation set data.

5.2. Locally weighted regression

Different locally weighted regression (LWR) models were investigated using the training set data to predict the stop set data to validate the model parameter choices. As opposed to the other models used in this study, these models were made to classify only "peak" versus "non peak". A binary classification was chosen for practical reasons as the software did not allow multiple classes. The final LWR model, predicting the validation set data, had 11 components and 400 points in the local regression.

5.3. ANN

Shallow ANN models classifying samples into the four classes were trained using the training data under the same conditions as the deep convolutional network, using the same neural networks toolbox in matlab2018a and the same learning parameters. A single hidden layer was used, while the number of neurons were varied from one to ten. The final ANN had two neurons in its hidden layer.

6. Results and discussion

In Fig. 5, the resulting ROC curves for the peak versus non peak classification of the validation data are shown for the four tested models. Each curve shows the classification performance as a function of the decision threshold. For example, it is possible to have perfect classification of true peaks (true positive rate) by setting the threshold low. The downside of that is that all non-peaks will also be classified as peaks (false positive rate). By adjusting the threshold, the ROC curve shows the performance compromise. The ideal curve would be in the upper left corner and have an area under the curve (AUC) of one. A completely nonfunctioning model would have a ROC curve lying on the diagonal from lower left to upper right and have an AUC of a half. Even though the deep network is trained to classify four different classes, it is the correct identification of peaks that is essential for the intended application, and so all models were evaluated based on this.

As is evident from the results, all models are capable of classifying reasonably well, but the deep learning net has better AUC than any of the other methods. The one that comes closest in performance is LWR. This is to be expected as it only builds models locally and hence can build models that do not need to consider shapes completely different from the one investigated. In PLS-DA and ANN, the classification is basically based on global latent variables which means that the classification model will trigger on *a* specific type of shape. Therefore, it will be difficult to handle extreme shifts in time axis position or extreme variations in width unless a very high false positive rate is accepted. The deep learning and LWR approaches, on the other hand, need not have one signal that will trigger positive. Because of the localized nature of both, they can handle much more heterogenous classes which is exactly what is needed here.

That local models perform well also implies that e.g. support vector machines could be a viable approach, but both those as well as LWR are very slow in practice for data sets of this size. Support vector machines would most likely be cumbersome to implement with hundreds of thousands of samples. When screening through the results, it is clear that for well-performing models and intervals, the deep network performs extremely well and hardly ever misses a peak. Some of the alternative methods can fail even for seemingly simple problems (see Fig. 6).

Nevertheless, it is evident from Fig. 5 that the deep network is not perfect. In order to understand why the classification results are not perfect, we manually went through all cases where the deep network did not get the classification of a peak or non peak correct.

In around half of the few cases where the deep network failed, the apparent error was caused by a bad reference class which means that our initial class assignment was wrong. In less than five percent of the misclassified cases, the deep network really did do worse than the alternative methods but mostly by confusing a peak with the category "other". The category "other" is the most heterogenous class and predominantly occurs in intervals and models that are already doing a fairly bad job. Hence, it is plausible that many of these models are deemed invalid on other grounds. Apart from these two types of errors constituting a little more than half the problems, the remaining misclassifications seem to be cases where the deep network makes mistakes but much less so than the competing models. Hence, overall, the deep network is trustworthy and efficient and even to a higher degree than the ROC curve seems to indicate. Training the model on more data of diverse nature may help improve the model even further.

As a final illustration of the use of the network, we show the outcome of the PARAFAC2 modelling of one interval in Fig. 7. As can be seen, the elution profiles are correctly assigned to the appropriate classes. Notice that the baseline-like profiles in component five are assigned as "other" because they are slightly negative which is inconsistent with how baselines are defined.

7. Conclusions

It has been shown that deep learning provides an approach for automatically assessing the nature of estimated elution profiles. With the well performing classification model implemented here, it is possible to further automate the analysis of chromatographic data. The current system was developed solely using GC-MS data, but it is of interest to test out the model on other types of chromatographic systems as well. It is also of interest to expand the current model to include other types of information; e.g. for detecting fronting and tailing. If such capabilities are included, the model may further help the chemist in providing advice and guidance.

Acknowledgments

The research in this paper was partly funded by The Nils Foss Excellence Prize awarded to one of the authors.

References

[1] L.G. Johnsen, P.B. Skou, B. Khakimov, R. Bro, Gas chromatography – mass spectrometry data processing made easy, J. Chromatogr. A 1503 (2017) 57–64.
[2] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, (2015) arXiv preprint arXiv:1502.03167, 2015 - arxiv.org.
[3] R.H. Hahnloser, R. Sarpeshkar, M.A. Mahowald, R.J. Douglas, H.S. Seung, Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit, Nature 405 (6789) (2000) 947–951.
[4] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City CA, 1991.
[5] W. Rawat, Z. Wang, Deep convolutional neural networks for image classification: A comprehensive review, Neural Comput. 29 (2017) 2352–2449.
[6] D.P. Kingma, J.L. Ba, Adam: A Method for Stochastic Optimization ICLR 2015, (2015) arXiv preprint arXiv:1412.6980, 2014 - arxiv.org.