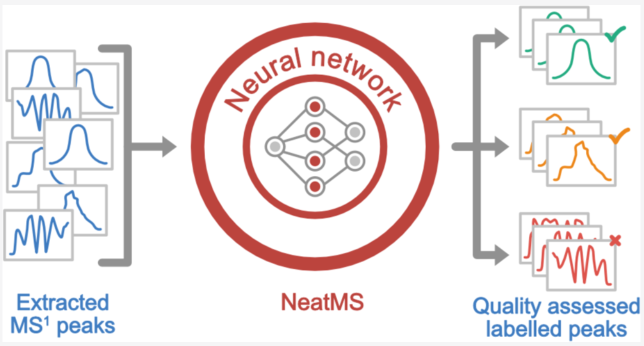
Deep Learning-Assisted Peak Curation for Large-Scale LC-MS Metabolomics

Yoann Gloaguen, Jennifer A. Kirwan, and Dieter Beule*

Cite This: *Anal. Chem.* 2022, 94, 4930−4937 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Available automated methods for peak detection in untargeted metabolomics suffer from poor precision. We present NeatMS, which uses machine learning based on a convoluted neural network to reduce the number and fraction of false peaks. NeatMS comes with a pre-trained model representing expert knowledge in the differentiation of true chemical signal from noise. Furthermore, it provides all necessary functions to easily train new models or improve existing ones by transfer learning. Thus, the tool improves peak curation and contributes to the robust and scalable analysis of large-scale experiments. We show how to integrate it into different liquid chromatography−mass spectrometry (LC-MS) analysis workflows, quantify its performance, and compare it to various other approaches. NeatMS software is available as open source on github under permissive MIT license and is also provided as easy-to-install PyPi and Bioconda packages.



■ INTRODUCTION

Liquid chromatography−mass spectrometry (LC-MS) is a widely used method in untargeted metabolomics. The postacquisition raw data processing, which aims to detect compound-related peaks and distinguish them from noise signals, is still a major challenge. Noise is defined as the background level of signal, which surrounds the chemical signals of interest. It can be caused by random fluctuations in the electrical environment, although its definition is often expanded to include chemical signals, which are of low intensity and quality and poorly reproducible. Many algorithms and tools have been developed to automate the process of peak picking (e.g., XCMS,[1] MZmine2,[2] Optimus,[3] MS-DIAL[4]). Pipelines for the automatic LC-MS raw data processing usually consist of the following steps: definition of regions of interest (ROI), detection of chromatographic peaks, quantification of these peaks, peak matching or grouping for samples within the batch or analysis, and clustering of peaks belonging to the same compound. XCMS and MZmine2 (called MZmine in the rest of the manuscript) are the most widely used open-source software that perform all of these steps and provide the user with a table of peaks found in the spectra and their integral intensities for each sample. However, there is a tendency for peak picking software to overpick peaks, i.e., they accept a large number of peaks, which are either (i) noise or (ii) low intensity, poorly reproducible peaks, or (iii) real peaks, which have incorrectly defined boundaries, thus creating a high number of false-positive peaks in the final dataset.[5] A previous detailed comparison of XCMS and MZMine demonstrated that the majority of peaks picked by both software packages were false positives.[6] Poor consistency between software is another major issue, with individual datasets showing overlaps of as little as 36% of the peaks picked by a single software.[6] Poor peak picking may obstruct or impede downstream analysis and biomedical interpretation of metabolomics studies and thus some kind of manual peak curation is still the norm. This also makes analysis of large-scale studies extremely laborious and limits reproducibility of analysis. Recent progress in machine learning (ML) algorithms[7] and availability of affordable parallel processing hardware (GPUs) has sparked the application of deep learning methods in both gas chromatography−mass spectrometry (GC-MS)[8,9] and LC-MS in peak detection.[10,11] ML has also been used for intra- and interbatch corrections[12] in LC-MS. A recent review[13] discusses the latest advancement of machine learning tools for LCMS-based metabolomics. Peakonly[10] is a deep neural network-based peak picking, segmentation, and integration algorithm that attempts to achieve a higher precision than conventional peak picking algorithm. However, it falls short in sensitivity compared to these tools. DNN[11] introduces the idea of postprocessing results from conventional algorithms with a neural network-based peak quality classification for the special

Received: May 27, 2021
Accepted: January 25, 2022
Published: March 15, 2022

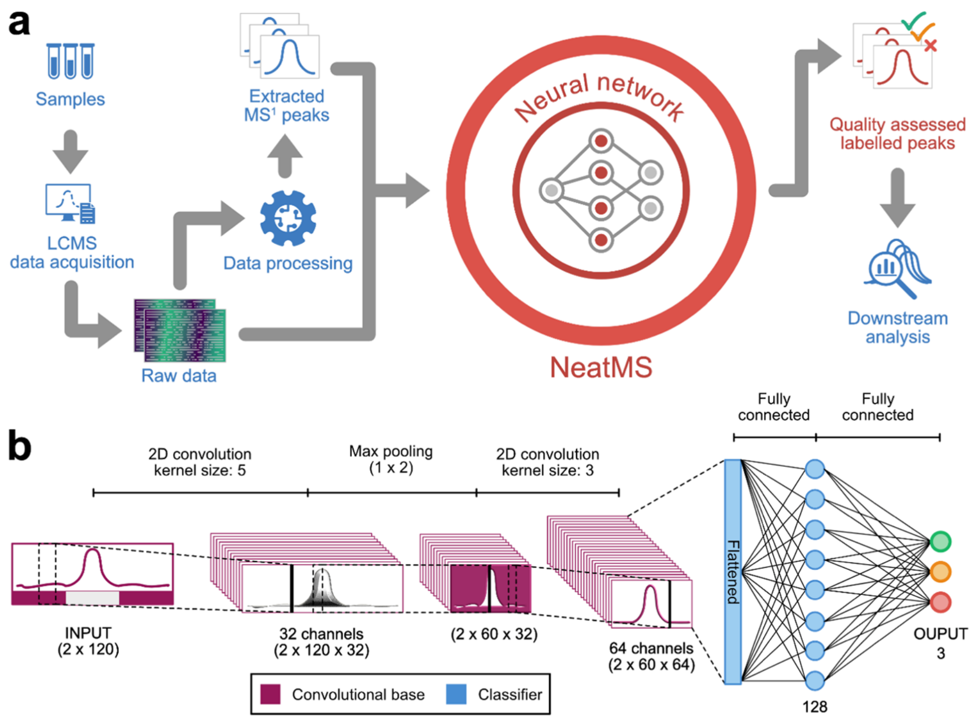


Figure 1. (a) Integration of NeatMS (red) into an existing workflow (blue): samples are run through the LCMS data acquisition, raw data are processed using standard automated tools to extract peaks, and both raw data and extracted peaks are used as input for NeatMS that assigns them to three classes: high quality, and acceptable or noise. The classification information can be used in the downstream analysis. NeatMS comes with a pre-trained network and includes all components for re-training and transfer learning. (b) Architecture of the neural network: NeatMS includes a two-dimensional (2D) convolutional base for feature extraction and a classifier made of two fully connected layers. This architecture was chosen due to its high performance in object detection and pattern recognition.[13] The max pooling layer between the convolutional layers reduces data size in the retention time dimension. This enables a higher abstraction of the data and reduces the number of learned parameters and thus helps to prevent overfitting as well as to reduce computational training effort. The classifier is made of two fully connected layers and uses a softmax function to produce three output values, which correspond to the peak classes.

case of prealigned peaks. This is an important proof of concept, but is not applicable to all study designs, and the required run time seems prohibitive for large-scale studies and routine use.

■ METHODS

Here, we introduce NeatMS, which is designed to serve as an independent deep learning-based peak filter tool in existing analysis pipelines. It addresses the overpicking issue by automatically evaluating and classifying peaks based on quality. To achieve this, we introduce three peak quality classes (high, acceptable, and “poor quality or noise”—henceforth called noise; see Figure S1 for some examples) and provide a pre-trained neural network to allow for out-of-the-box usage. Transfer learning and complete re-training are also supported. NeatMS can be easily integrated into existing workflows (see Figure 1a); it uses a convolutional network architecture shown in Figure 1b.

NeatMS is designed to be easily integrated into existing workflows and is adaptable to different measurement protocols, instruments, and preprocessing tools. Therefore, NeatMS can be used in different ways. Most simply, it is used to classify quality of an input dataset using an existing model. NeatMS also allows the creation of new or improvement of existing classification models using training data. The training data can be generated with an integrated labeling tool. For all usage, the input data formats are the same and processing always starts with a data preparation step.

LC-MS Data Acquisition. To evaluate the performance of NeatMS, we use two datasets with known chemical standards (CS). Dataset 1 consists of 20 quality control samples “QC 2” from the Biocrates P400 kit[14] consisting of 80 chemical standards (CS) at proprietary concentrations in a lyophilized plasma matrix. Dataset 2 is based on the Biocrates kit calibration sample “Cal 1”, which is matrix-free and contains 41 of the chemical standards also present in dataset 1. Note however that the concentration of the chemical standards (CS) in “Cal 1” is substantially lower than that in “QC 2”, and the detailed concentration relation is however an unknown trade secret of Biocrates. We created a serial dilution in water (1:1.2, 1:1.4, 1:1.6, 1:1.8, 1:2, 1:3, 1:5, 1:7.5, 1:10, 1:15, 1:20, 1:30, 1:50, and 1:100). Following dilution, we added 39 compounds (Biocrates internal standard mix) at the same concentration to each sample to act as chemical background. Each dilution was measured in triplicate. The objective of this dilution series was to quantitatively assess how NeatMS performed over a range of low peak intensities and near the limit of detection. All data were acquired following Biocrates P400 kit standard protocol on an Agilent 1290 coupled with a Thermo Q Exactive instrument. Datasets 1 and 2 were preprocessed with MZmine and XCMS using the versions and parameters detailed in Supporting Information Table S1. For XCMS centWave, we applied IPO[15] for parameter optimization individually for each dataset; furthermore, we used the XCMS peak merging feature. For MZmine, we used the parameters recommended in the user documentation.

Data Preparation. Although the peak detection is performed by an external tool, workflow or pipeline, the full signal is used for classification and is directly retrieved from the raw data. This prevents biases originating from data transformations applied by the different peak detection tools (baseline correction, smoothing, etc.). Therefore, the input data of the module consists of .csv formatted files describing the peaks detected and the raw sample files in mzML format. Other vendor-specific raw file formats can be converted into mzML format using the msconvert tool available in ProteoWizard.[16] The csv input files can be generated using standard preprocessing tools such as MZmine or XCMS. The position of the module within standard data processing workflows is illustrated in Figure 1a. The output of NeatMS is again in csv format and contains the information from the input csv as well as the peak classification and labeling generated by the software.

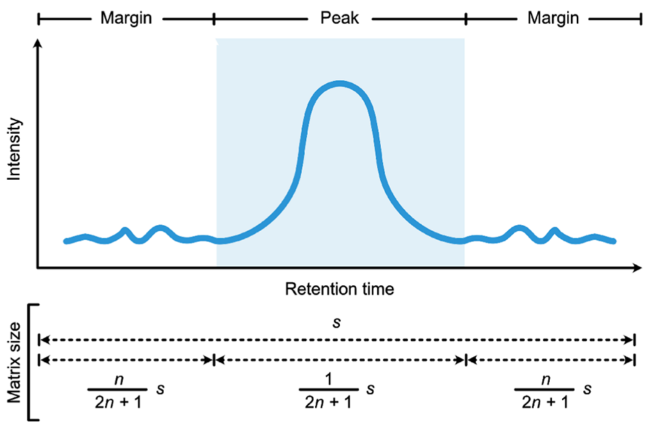
Before any transformation, NeatMS excludes unacceptably narrow peaks from further processing by requiring a minimum scan number of 5 (configurable minimum scan number input filter). As the model evaluates the peak shapes and the quality of their extraction from the raw signal (e.g., peak boundaries), it is important to provide contextual information. This is performed by extracting a larger retention time (RT) window containing both the peak itself, as defined by the preprocessing tool, and the signal surrounding the peak (called peak margin thereafter). The RT window of the signal to be extracted is defined as follows (with n = 1 by default)

$$\begin{aligned}\text{RT window} \in (&\text{peak start RT} - n \times \text{peak width}\\ &, \text{peak end RT} + n \times \text{peak width})\end{aligned}$$

A min−max normalization is then applied to the extracted signal

$$\frac{x - \min(x)}{\max(x) - \min(x)}$$

The resulting signal is linearly interpolated to obtain a vector of size s (120 by default). The $s/(2n + 1)$ that is, by default, the central 40 values represents the peak signal, and the $n \times s/(2n + 1)$ that is by default the 40 values on each side represents the peak margins as shown in Figure 2. A second (binary) vector of



I	0.002	0.003	0.45	1.0	0.42	0.008	0.01
W	0	0	1	1	1	0	0

Figure 2. Data structure of a single peak after preprocessing for neural network feeding; s: size of scaled intensity vector, n: margin width (as a fraction of peak width), I: scaled intensity vector of size s, W: binary window vector of size s (0 = margin, 1 = peak).

length s is then created to describe whether an intensity value (single point) is part of the peak window (1) or the margin (0). The resulting data structure is a two-dimensional tensor, or matrix, of size $2 \times s$, as shown in Figure 2. Although n and s can be adjusted by the user, the pre-trained model provided with NeatMS has been trained using default parameters and requires no adjustment when using this model.

Convoluted Neural Network Architecture. The neural network (see Figure 1b) is built following generic convolutional network architecture including a convolutional base for feature extraction and a classifier made of fully connected layers. The convolutional architecture of the network was selected due to its high performance in object detection and pattern recognition.[17] The convolutional base is composed of two convolutional layers, with a max pooling layer between them. This operation halves the size of the data in the retention time dimension and enables a higher abstraction of the data to classify, which helps to prevent potential overfitting. This layer also reduces the number of parameters to learn and the computational cost. The classifier part of the network is made of two dense layers and produces three output values through a softmax activation function, which corresponds to the number of peak quality classes. Rectified linear unit (ReLU) activation function is used throughout the rest of the network. The convolutional layers use a stride of 1 with a "same" padding. The kernel size and channel number for every layer are detailed in the figure.

Training. Our NeatMS analysis first evaluates the pre-trained (PT) model. This initial model was trained by the authors on a wide range of peak shapes from different datasets. Additionally, we used the transfer learning approach to optimize the generic PT model to the specifics of dataset 1; we will call the resulting second model transfer learning (TL). The pre-trained (PT) as well as the transfer learning (TL) model used in this work are provided in the Supporting Information. All components needed to perform creation of new models or use transfer learning to improve existing ones are part of the open-source NeatMS software. Thus, users can also retrain the system for their specific column/instrument/peak detection workflow combination if the provided pre-trained (PT) model does not suffice.

The creation of a new training dataset is facilitated by an interactive visualization and labeling tool that can be run within a Jupyter notebook.[18] Preparing a training dataset requires the user to be experienced with mass spectrometry data analysis and chromatographic peaks evaluation. The decisions by the user for the training data will be learned by the algorithm and thus must be consistent and trustworthy. This tool requires the same input as the main NeatMS tool and presents the user with randomly selected peaks for manual assignment to the three labeling classes. Typically, a few hundred peaks can be labeled within an hour. The PT model is based on about 5000 peaks, and the TL model was adjusted using about 2500 peaks. Once the desired number of peaks have been labeled, the model can be trained using two different approaches (full training or transfer learning, see below). The labeled dataset is divided into an 80/10/10 training/test/validation split by default. Model testing and validation are always performed the same way regardless of the training approach chosen. The test set is used during the actual training process to prevent overfitting. The validation set remains untouched during the entire training process and is subsequently used for hyperparameter optimization. This

Table 1. NeatMS and Peakonly Comparison[a]

		NeatMS MZmine TL model	NeatMS MZmine PT model	NeatMS XCMS TL model	NeatMS XCMS PT model	peakonly
peak number	input	6977	6977	5994	5994	1907
	classified	5505	5505	4513	4513	not reported
	high quality	1069	2127	2280	2635	not reported
	acceptable quality	1945	1817	714	1088	not reported
	noise	2491	1560	1519	791	not reported
CS found	input and classified	94.25%	94.25%	97.13%	97.13%	79.44%
	high quality	79.31%	88.19%	91.44%	92.94%	not reported
	acceptable quality	11.25%	4.94%	2.13%	3.69%	not reported
	noise	3.69%	1.13%	3.56%	0.50%	not reported

[a]Tool and model comparison using dataset 1 showing average number of peaks found across 20 samples and average percentages of detected CS. The input row shows the results returned by the original peak detection tools (MZmine, XCMS), and other rows show the details of the three peak classes given by NeatMS. The total number of peaks after classification is smaller than the input due to the application of a minimum scan number filter that NeatMS uses (a default value of 5 is used).

optimization can be performed automatically or manually, but performing it manually allows more control over the specificity vs. sensitivity of the model. Instructions for the manual process are given in the documentation.

The training tool enables the freezing of any network layer, making it possible to select the specific layers in which weights should be adjusted. It is, however, considered better practice to only adjust the classifier part by freezing the entire convolutional base when the training set is very small. Guidance on layer selection is given in the advanced section of the documentation. This approach is especially important for transfer learning and enables fine tuning of the pre-trained models by further training specific layers.[19] The advantage of this approach over full training is that it requires a much smaller training dataset and thus less manual labeling effort. However, the tool also supports full training of entirely new models. This approach consists of using only the network architecture and fully training the model from scratch. This approach will produce the best results for the data being analyzed but requires a large training dataset. Instructions on how to import models are available in the documentation.

Implementation. NeatMS is written in Python 3.6 and is available as a python package through PyPi package installer and Bioconda. The data handling and operations are performed using NumPy[20] and scikit-learn,[21] and the neural network is constructed using Keras[22] and Tensorflow.[23] As a Python package, the intended use of the module is to be embedded as an extra step within a data processing pipeline. The module can be integrated and automatically executed by any pipeline or workflow management tool capable of running python code. However, it can also be used as a standalone application through a dedicated python script or within a Jupyter notebook. Several Jupyter notebooks are provided for tutorial purposes and can serve as templates and examples. The generated results are reported in standard .csv format and can also be exported as pandas[24] dataframes for direct integration in python-supported pipelines. The structure of the output can be controlled through a dedicated method to ensure smooth integration into the majority of data processing pipelines. Optional filters can also be turned on and parameterized. Details about the full usage of the export method are provided in the documentation.

RESULTS AND DISCUSSION

Table 1 summarizes the number of peaks and class assignments from NeatMS for dataset 1 using both the PT and TL models. Table 1 shows that even if we do not use transfer learning, NeatMS can still deliver a useful improvement for existing MZmine workflows. It substantially reduces the number of peaks that need to be considered for downstream analysis. This facilitates, e.g., differential analysis either on only high-quality data or on high- and acceptable-quality data. For XCMS, the separation between acceptable and high-quality class is not so clean due to the fact that the training set was exclusively based on peaks reported by MZmine. Both MZmine and XCMS users can start working with the PT model and will immediately improve their workflow performance and incrementally create further improvements by training better models. Further discussion will be focused on NeatMS using the MZmine trained TL model. Results with XCMS could likely be further improved by creating an XCMS-specific trained model.

Median relative standard deviations (RSDs) increase substantially from high-quality to noise peaks (Figure 3b), with acceptable-quality peaks falling in between. NeatMS quality class assignment differs strongly from conventional QC-RSD filtering methods because we also observe many noise peaks with low RSDs. This effect is present with XCMS and MZmine for both models (not shown). Figure 3d,e shows that the CSs are consistently found by NeatMS across various low dilution samples and generally tend to move from high to acceptable quality as dilution increases. Eventually, some fall into the NeatMS noise category while most can no longer be detected by MZmine as the signal decreases with increasing dilution. Figure S2 (peak width distribution) shows that the noise class is dominated by rather broad peaks while the high-quality class shows a consistent peak width distribution independent of the peak area. This indicates that our three classes represent a sensible quality classification for peaks. Figure 3a (ROC curve) shows that the learning itself was very successful, the model closely resembles the expert knowledge of the trainer. Thus, NeatMS evaluation is comparable to, but much faster and much more reproducible and consistent than, human expert evaluation. NeatMS must be considered superior for large-scale studies with hundreds or thousands of samples and potentially several millions of peaks. By including training and transfer learning functionality into our solution, we empower researchers to adapt and optimize the learned

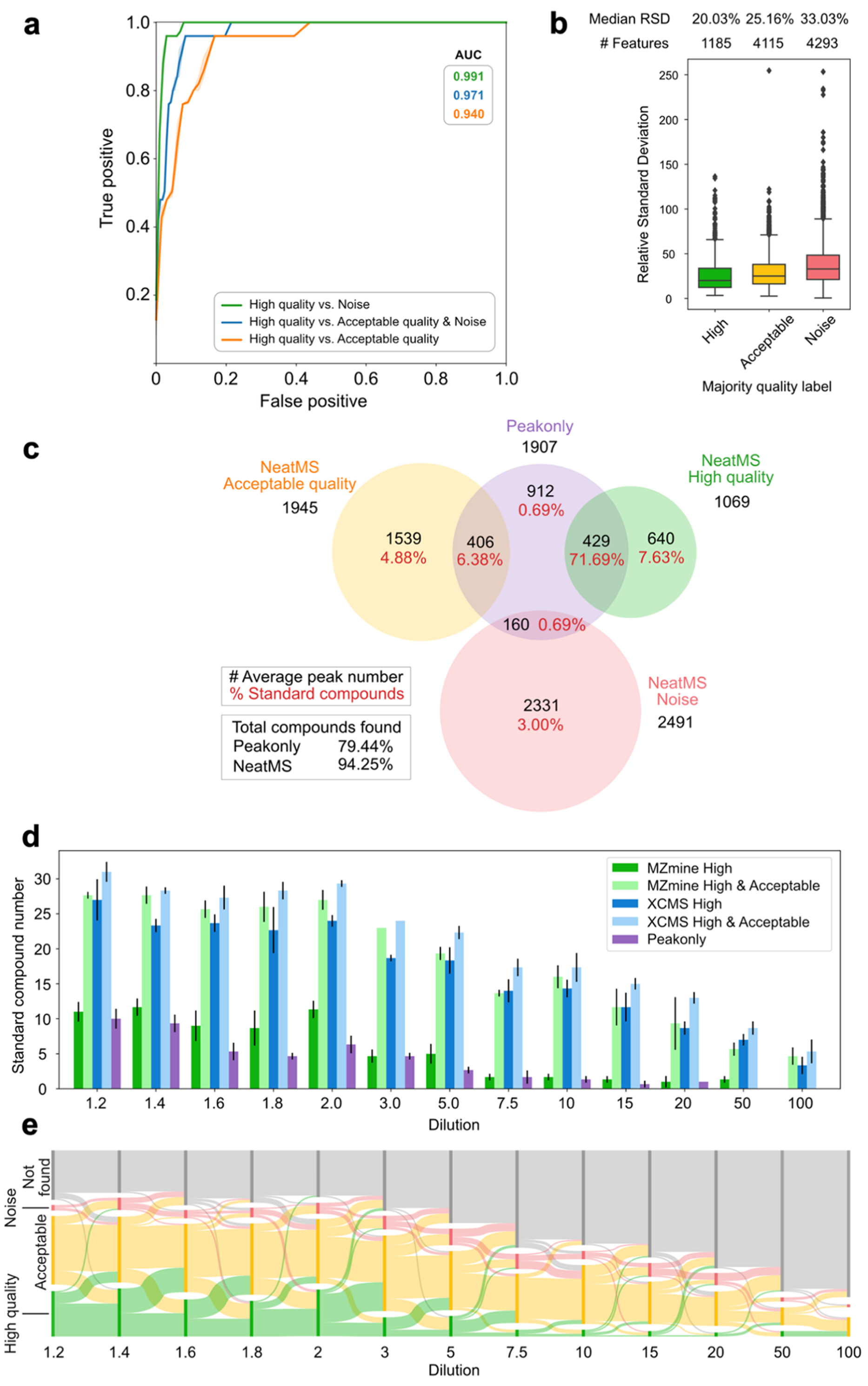


Figure 3. NeatMS results for dataset 1 (a−c) and dataset 2 (d, e) based on TL model and MZmine output. (a) ROC curves showing learning efficacy for three different group separations. The curves are created using the validation set of the TL model learning; the high-quality class probability returned by the model is used as the varying parameter. (b) Box plots of RSDs: high-quality peaks show lower individual RSDs and reduced within-class RSD variability. Each feature for which peaks were present in at least four samples was assigned to a quality based on the most frequently reported class. (c) Venn diagram comparing peakonly and NeatMS. Numbers show averages over the 20 samples: total number of detected peaks (black), percent of recovered CSs (red). On average, NeatMS reports 63.5 CSs as high quality, i.e., recovers 79% of original, another 11% are considered acceptable quality. Note that matching peaks derived by different algorithms is challenging in itself and not completely unambiguous; cf. Data Analysis Details section for details. (d) Classification of individual CS over different dilutions and for different tools, NeatMS high-quality class outperforms peakonly for most dilutions. A substantial number of additional CSs are categorized in the "acceptable quality" class. (e) Sankey diagram showing the distribution of the 41 diluted CSs for all three replicates in different quality classes and their change between dilution steps. Each dilution step is represented by a stacked barplot; the widths of the flows between bar plots (dilutions) represent the fraction of CSs going from one class to another.

classification and filtering to their specific data, needs, and preferences.

Data Analysis Details. Our validation with dataset 1 focuses on the number of known spiked chemical standards found by the different tools. Peak identification was performed using compound-specific RT and *m*/*z* tolerance windows provided with the Biocrates kit. To compute the final results, percentages were calculated on the basis that all 80 compounds are detectable in all 20 samples. The same approach was used to analyze the recovery of CSs in dataset 2. These samples contain the same 80 CSs as dataset 1, but only 41 of the CSs were diluted, while 39 were used as internal standards and thus were at the same concentration throughout (Figure S3b). The Sankey diagram (Figure 3e) was created by comparing the peak classes of the CSs for all three replicates in the consecutive dilution points to generate migration flows. A class corresponding to nondetected CSs was added to conserve an even CS population size throughout. To create the Venn diagrams (Figures 3c and S5), peaks reported by one tool were compared to the list of peaks reported by the other tool and considered the same when any two peaks presented a mutual overlap higher than 50% in the RT and *m*/*z* dimensions, respectively. However, tools can differ widely in the peak boundaries assignment for the same peak. Therefore, any peak matching method will remain imperfect and ambiguous. This explains the noncomplete overlap of peaks found by NeatMS compared to peakonly. To generate RSD results in Figure 3b, we used peak alignment (using the MZmine join aligner algorithm) and assigned the features to the most frequently found quality class across the 20 samples. Features were retained only when present in a minimum of four samples.

Comparison with Peakonly. Table 1 also shows a comparison of NeatMS to results obtained with the recently published peakonly tool,[10] which also applies machine learning on raw data to detect high-quality peaks. We choose the peakonly parameters as suggested by the authors and also tried to further optimize them; see Figure S4. Unfortunately, it was not possible to apply transfer learning or any re-training for peakonly because the software does not provide the necessary components to do this.

Figure 3d shows that NeatMS high-quality class equals or outperforms peakonly in CS peak recognition for all dilutions. Additional CSs are classified as acceptable. A more detailed comparison for dataset 1 is shown in Figure 3c (for XCMS, we find comparable results; see Figure S5). NeatMS High and Acceptable quality classes together contain on average more than 90% of CSs. Peakonly reports an average of 1907 peaks, containing on average 79% of the CSs. Approximately the same percentage of CS is contained in 1069 high-quality peaks reported by NeatMS. The concordance is however not perfect. Only rarely do we miss CSs that peakonly reports (0.7%). A small portion of the CS matched signals are considered Noise by NeatMS. Upon visual inspection, our expert usually agrees with the NeatMS algorithm (see Figure S6).

Comparison with DNN and MetaClean. In addition to the comparison with peakonly,[10] we also compared NeatMS with a deep neural network-based peak filtering tool developed by Kantz et al.[11] (referred to as DNN tool thereafter) using their default parameters. In contrast to peakonly and NeatMS, DNN uses a peak position imputation/gap filling approach, i.e., positions that contain peaks in a certain minimal number of samples are evaluated and quantified in all samples. Similar functionality is also optionally provided by MZmine; thus, NeatMS can also support this approach although this makes an unbiased evaluation of tool performance more difficult. This becomes very obvious in dataset 2, where we do not want to use information from the highly concentrated samples to infer algorithm performance on the lowly concentrated samples. Figure 4a visualizes the comparison of NeatMS and DNN for dataset 1, and the full confusion matrices are shown in Supporting Information Tables S4 and S5. Note that gap filling does not substantially alter NeatMS results; cf. Table S3. Overall, DNN finds about 85% CS in the Good category compared to about 91% by NeatMS. Overall, the results of the two tools are consistent but NeatMS still outperforms DNN. Furthermore, processing with DNN required about 24 h, while NeatMS took less than 10 min on the same hardware. Similar differences applied in terms of storage needs. This is in line with the DNN authors claiming to provide an "important proof of concept" and our ambition to provide easy-to-use software. Users of all three tools—DNN, peakonly, or NeatMS—should be aware that just as any other compound, contaminants may produce high-quality or low-quality peaks. Addressing contaminant-related issues is indeed an important task but is outside of the scope of this work or tools like DNN or peakonly, respectively.

Finally, we compared NeatMS performances with those of MetaClean,[25] which uses a machine learning-based classifier to

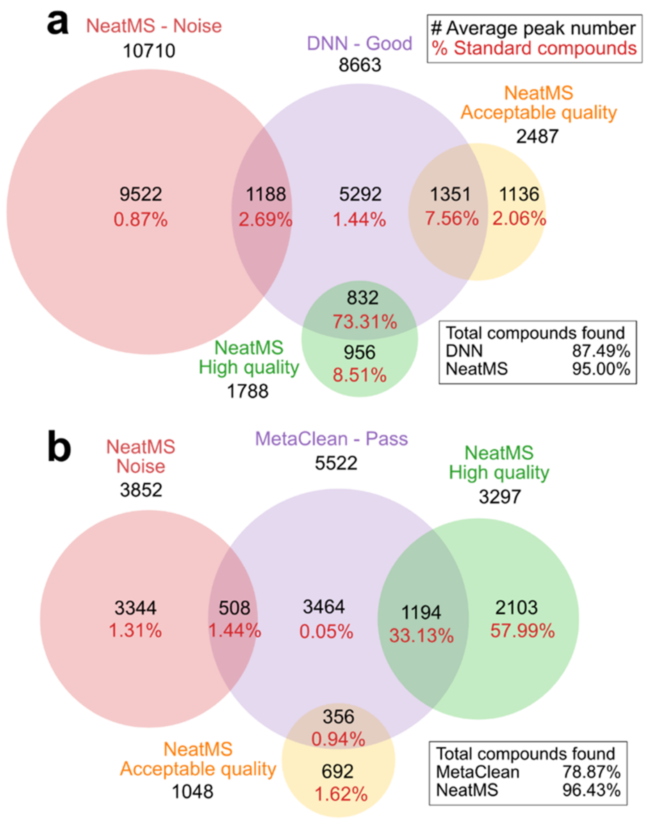


Figure 4. Venn diagram comparing (a) DNN and NeatMS and (b) MetaClean and NeatMS analogue to peakonly comparison in Figure 1c using dataset 1. Numbers shown are averages over 20 samples: total number of detected peaks (black), percent of recovered CSs (red). The total compounds percentage refers to the total recovered CS. For clarity, the figure is limited to MetaClean "Pass" and DNN "Good" classes, and the full confusion matrices are given in Supporting Information Tables S4–S7.

reduce false-positive peak detection by assigning "Pass" or "Fail" labels to peaks aligned and grouped across samples. The tool is designed to use with XCMS, and peaks must be aligned and gap-filled. Figure 4b visualizes the comparison of NeatMS and MetaClean for dataset 1, and the full confusion matrices are given in Supporting Information Tables S6 and S7. At least for this dataset NeatMS recovers much more CS (about 94%) compared to 36% by MetaClean. Compute and storage resources requirements are comparable to NeatMS.

CONCLUSIONS

NeatMS outperforms existing tools for peak curation; it requires neither large computing power nor long computing time; all data analysis described in this manuscript can be done with a standard laptop within minutes, and all described training can be done with a modern PC within a few hours. NeatMS software is available as open source on github under permissive MIT license and also provided as easy-to-install PyPi and Bioconda[26] packages. NeatMS comes with comprehensive user documentation, tutorials, and importantly also contains an easy-to-use training tool. Users can thus create their own models or improve existing ones according to their specific needs. NeatMS supports standard input and output formats and is therefore easily added into existing workflows. Thus, it is compatible with many use cases and may help to enable improved and reproducible data analysis for large LC-MS studies.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.analchem.1c02220.

Example of CS peaks (Figure S1); peak width density plots per quality class (Figure S2); peak number and internal standard recovery (Figure S3); average detected peak number and standard compound recovered by peakonly on dataset 1 relative to parameter selection (Figure S4); Venn diagram comparing peakonly and the combination XCMS with NeatMS TL model (Figure S5); extracted ion chromatogram of a noise peak reported by MZmine (Figure S6); parameters of the different tools used for datasets 1 and 2 (Table S1); performance of peakonly with NeatMS (Table S2); NeatMS results using TL model on dataset 1 (Table S3); confusion matrix of DNN and NeatMS number of peaks per class (Table S4); confusion matrix of DNN and NeatMS CS recovery per class (Table S5); confusion matrix of MetaClean and NeatMS number of peaks per class (Table S6); and confusion matrix of MetaClean and NeatMS CS recovery per class (Table S7) (PDF)

AUTHOR INFORMATION

Corresponding Author

Dieter Beule − *Berlin Institute of Health at Charité, Core Unit Bioinformatics, 10178 Berlin, Germany; Max Delbrück Center for Molecular Medicine in the Helmholtz Association, 13125 Berlin, Germany*; Email: dieter.beule@bih-charite.de

Authors

Yoann Gloaguen − *Berlin Institute of Health at Charité, Metabolomics Platform, 10178 Berlin, Germany; Berlin Institute of Health at Charité, Core Unit Bioinformatics, 10178 Berlin, Germany; Max Delbrück Center for Molecular Medicine in the Helmholtz Association, 13125 Berlin, Germany*; orcid.org/0000-0003-0493-8592

Jennifer A. Kirwan − *Berlin Institute of Health at Charité, Metabolomics Platform, 10178 Berlin, Germany; Max Delbrück Center for Molecular Medicine in the Helmholtz Association, 13125 Berlin, Germany*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.analchem.1c02220

Author Contributions

The manuscript was written through contributions of all authors. All authors have given approval to the final version of the manuscript.

Notes

The authors declare no competing financial interest.
The github repository contains some sample data from dataset 1; the full datasets can be downloaded from http://doi.org/10.5281/zenodo.3973172.
NeatMS is open source and is freely available at https://github.com/bihealth/NeatMS under permissive MIT license. A PyPi package is available at https://pypi.org/project/NeatMS/; a Bioconda package is available at https://anaconda.org/bioconda/neatms. The user documentation can be found at https://neatms.readthedocs.io/en/latest/.

ACKNOWLEDGMENTS

The authors thank Alina Eisenberger and Raphaela Fritsche for generating datasets 1 and 2, Friederike Gutmann and Mathias Kuhring for testing NeatMS and providing valuable feedback, and Eric Blanc for his valuable insight on machine learning. This work was supported by BMBF-grant 031L0220A.

REFERENCES

(1) Smith, C. A.; Want, E. J.; O'Maille, G.; Abagyan, R.; Siuzdak, G. *Anal. Chem.* **2006**, *78*, 779−787.
(2) Pluskal, T.; Castillo, S.; Villar-Briones, A.; Orešič, M. *BMC Bioinf.* **2010**, *11*, No. 395.
(3) Protsyuk, I.; Melnik, A. V.; Nothias, L.-F.; Rappez, L.; Phapale, P.; Aksenov, A. A.; Bouslimani, A.; Ryazanov, S.; Dorrestein, P. C.; Alexandrov, T. *Nat. Protoc.* **2018**, *13*, 134−154.
(4) Tsugawa, H.; Cajka, T.; Kind, T.; Ma, Y.; Higgins, B.; Ikeda, K.; Kanazawa, M.; VanderGheynst, J.; Fiehn, O.; Arita, M. *Nat. Methods* **2015**, *12*, 523−526.
(5) Myers, O. D.; Sumner, S. J.; Li, S.; Barnes, S.; Du, X. *Anal. Chem.* **2017**, *89*, 8696−8703.
(6) Myers, O. D.; Sumner, S. J.; Li, S.; Barnes, S.; Du, X. *Anal. Chem.* **2017**, *89*, 8689−8695.
(7) Zhang, A.; Lipton, Z. C.; Li, M.; Smola, A. J. Dive into Deep Learning, *arXiv preprint* **2021**, arXiv:2106.11342.
(8) Borgsmüller, N.; Gloaguen, Y.; Opialla, T.; Blanc, E.; Sicard, E.; Royer, A.-L.; Le Bizec, B.; Durand, S.; Migné, C.; Pétéra, M.; Pujos-Guillot, E.; Giacomoni, F.; Guitton, Y.; Beule, D.; Kirwan, J. *Metabolites* **2019**, *9*, No. 171.
(9) Lebanov, L.; Tedone, L.; Ghiasvand, A.; Paull, B. *Talanta* **2020**, *208*, No. 120471.
(10) Melnikov, A. D.; Tsentalovich, Y. P.; Yanshole, V. V. *Anal. Chem.* **2020**, *92*, 588−592.
(11) Kantz, E. D.; Tiwari, S.; Watrous, J. D.; Cheng, S.; Jain, M. *Anal. Chem.* **2019**, *91*, 12407−12413.

(12) Rong, Z.; Tan, Q.; Cao, L.; Zhang, L.; Deng, K.; Huang, Y.; Zhu, Z.-J.; Li, Z.; Li, K. *Anal. Chem.* **2020**, *92*, 5082−5090.
(13) Rampler, E.; Abiead, Y. E.; Schoeny, H.; Rusz, M.; Hildebrand, F.; Fitz, V.; Koellensperger, G. *Anal. Chem.* **2021**, *93*, 519−545.
(14) AbsoluteIDQ p400 HR Kit - Metabolomics kit for Exactive^{TM}. https://biocrates.com/absoluteidq-p400-hr-kit/ (accessed July 28, 2020).
(15) Libiseller, G.; Dvorzak, M.; Kleb, U.; Gander, E.; Eisenberg, T.; Madeo, F.; Neumann, S.; Trausinger, G.; Sinner, F.; Pieber, T.; Magnes, C. *BMC Bioinf.* **2015**, *16*, No. 118.
(16) Chambers, M. C.; Maclean, B.; Burke, R.; Amodei, D.; Ruderman, D. L.; Neumann, S.; Gatto, L.; Fischer, B.; Pratt, B.; Egertson, J.; Hoff, K.; Kessner, D.; Tasman, N.; Shulman, N.; Frewen, B.; Baker, T. A.; Brusniak, M.-Y.; Paulse, C.; Creasy, D.; Flashner, L.; Kani, K.; Moulding, C.; Seymour, S. L.; Nuwaysir, L. M.; Lefebvre, B.; Kuhlmann, F.; Roark, J.; Rainer, P.; Detlev, S.; Hemenway, T.; Huhmer, A.; Langridge, J.; Connolly, B.; Chadick, T.; Holly, K.; Eckels, J.; Deutsch, E. W.; Moritz, R. L.; Katz, J. E.; Agus, D. B.; MacCoss, M.; Tabb, D. L.; Mallick, P. *Nat. Biotechnol.* **2012**, *30*, 918−920.
(17) Rawat, W.; Wang, Z. *Neural Comput.* **2017**, *29*, 2352−2449.
(18) Perez, F.; Granger, B. E. *Comput. Sci. Eng.* **2007**, *9*, 21−29.
(19) Pan, S. J.; Yang, Q. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345−1359.
(20) van der Walt, S.; Colbert, S. C.; Varoquaux, G. *Comput. Sci. Eng.* **2011**, *13*, 22−30.
(21) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. *J. Mach. Learn. Res.* **2011**, *12*, 2825−2830.
(22) Chollet, F.et al.*Keras*, 2015.
(23) Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015.
(24) McKinney, W. In *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference; van der Walt, S.; Millman, J., Eds.; 2010; pp 56−61.
(25) Chetnik, K.; Petrick, L.; Pandey, G. *Metabolomics* **2020**, *16*, No. 117.
(26) Grüning, B.; Dale, R.; Sjödin, A.; Chapman, B. A.; Rowe, J.; Tomkins-Tinch, C. H.; Valieris, R.; Köster, J. *Nat. Methods* **2018**, *15*, 475−476.