RAMAN SPECTROSCOPY DECONVOLUTION USING STACKED AUTO-ENCODERS WITH NON-NEGATIVITY CONSTRAINTS

Jacob S. Larsen, Flavia D. Frumosu, Jakob Thrane, Maximillian F. Vording, Tommy S. Alstrøm

Department of Applied Mathematics and Computer Science, Technical University of Denmark

ABSTRACT

The standard tool to analyze raman spectroscopy signal is non-negative matrix factorization (NMF). In some of the cases, the observed signal is not a linear combination of pure spectra, thus removing the basic assumption behind using NMF for analyzing these signals. The research question addressed in this paper is whether the use of non-negative sparse autoenconders are able to model the nonlinearities and thus are able to correctly deconvolve Raman Spectre. A model used for this purpose has been initially tested on the classical MNIST dataset with good results. A simulated raman spectra dataset has been created and the same model has been compared with the classical approach NMF. As the results indicate, NMF performs better than the proposed model. However there is a strong belief that a few changes in the model will improve the results significantly.

Index Terms— Raman spectroscopy, autoencoder, non-negativity constraints, non-negative matrix factorization, sparsity

1. INTRODUCTION

Raman scattering is a technique used to detect and identify molecules using the interaction of photons. It is a technique that is commonly used in chemistry and physics for detecting molecules by observing their vibrational and rotational modes. This is completed with the use of laser light. The photon excites the sample which causes scattering, meaning the photon has a change in energy over a short time period. This energy-change is reflected as a shift in frequency, also called a stokes shift, and by analysing the spectrum the molecules and combination of molecules can be identified.

Surface-enhanced Raman Spectroscopy (SERS) is an enhancement of Raman Scattering, that uses surfaces such as metal or nanostructures to absorb molecules. This enables the identification of single molecules. For instance, noble metal nanostructures can concentrate light which greatly enhances the electromagnetic field near the nanostructure. These areas become the so called "hot spots" that amplify weak Raman scattering signals. The placement and design of these nanostructers with high SERS preformance is beyond the scope of this work, however additional information can be found in [1] and references herein.

Modern SERS analysis is done with highly efficient nanostructures, and lasers with low linewidths resulting in very highly resoluted Raman intensity spectras. This provides very large complex matrices that contain patterns and structures. Multivariate curve resolution methods (MCR) and NMF (Nonnegative matrix factorization) have successfully been applied for matrix factorization [2]. These methods do however fail at very low signal-to-noise ratios and any meaningful spectral components are difficult to recover. This is moreover increasingly difficult when interactions create interactions that are not linear separable [3].

This work aims to evaluate the application of Sparse Autoencoders with non-negativity constraints on Raman spectra obtained from SERS. The novelty of this work consists of comparing traditional curve resolution methods such as NMF to the methodology and work completed by [4].

This work is organized as follows. Section 2 presents the methodology of non-negative matrix factorization and sparse autoencoders. Section 3 is a description of the non-negativity constraint in sparse autoencoders as proposed by [4]. Section 4 details the implementation done in TensorFlow and verification using the well known MNIST dataset. Moreover this section also details the SERS dataset used for the primary results. Section 5 describes the results obtained. A discussion is provided in Section 6 and a conclusion in Section 7.

2. METHODS

This work seeks to compare traditional and proven methods for SERS analysis, i.e. curve resolution, such as NMF to denoising autoencoders with non-negativity constraints.

2.1. Non-negative matrix factorization

Non-negative matrix factorization (NMF) consists of factorizing a original matrix V, with only positive elements, into two positive matrices W and H, [5].

$$V \approx W \times H \tag{1}$$

Where columns of W are considered basis vectors, and each column in H is considered an encoding with a one-to-one relationship with the columns of V. Thus, for instance, a matrix V of size $n \times m$ can be factorized into a matrix W of size $n \times k$ and a matrix H of size $k \times m$, where k is the number of components. Approaching the intuitive point of view, W and H can be seen as components that combined approximate the original signal V.

An iterative algorithm is considered for NMF, which shares similar monotonic convergence as the EM algorithm, [6]. Moreover, the rules of update preserve non-negativity of W and H. The algorithm approaches the problem by initialization of W and H as non-negative, and then update the values in W and H until local maxima is obtained and both matrixes are considered stable. The rules of multiplicative update can be defined as:

$$H^{n+1} \leftarrow H^n \frac{(W^n)^T V}{(W^n)^T W^n H^n} \tag{2}$$

and

$$W^{n+1} \leftarrow W^n \frac{V(H^{n+1})^T}{W H^{n+1}(H^{n+1})^T} \tag{3}$$

This, however, leads to an optimization that is convex for W and H, which in turn can require many iterations and result in poor local minima. For this problem, alternating least squares (ALS) is commonly used. This consists by alternating the optimization between W and H, e.g. one interation consists of 1) keeping H fixed while minimizing least squares for W, and 2) keeping W fixed while minimizing least squares for H. Further details can be found in [7] and references herein. ALS is for the remainder for this work, assumed the default optimization method when mentioning NMF.

2.2. Sparse auto-encoder

A regular auto-encoder seeks to learn a hidden representation that makes it possible to reconstruct the original input, [8]. The mapping from an input vector to a hidden/latent representation, the encoding step, can be described as:

$$y = f(x) = \sigma(\mathbf{W}x + \mathbf{b}) \tag{4}$$

Where x is the input vector and $\mathbf{W}$ is a matrix of weights and $\mathbf{b}$ is a bias vector. σ is an element-wise logistic sigmoid. y is the latent representation. From the latent representation it is then mapped back, i.e. reconstructed, in input space and can be described as:

$$z = g(y) = \sigma(\mathbf{W}'x + \mathbf{b}') \tag{5}$$

Where z can be seen as the reconstructed vector, and the weight matrix W' is the reverse mapping. The weights at both the encoding and decoding layer are to be optimized in order to minimize a "reconstruction error", thus the approach here is unsupervised. This can be formulated as the average reconstruction error using the squared error as a loss function

$$J_E(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_n^N ||x^i - z^i||^2 \tag{6}$$

Where N is the number of training samples. By limiting the size of the hidden layer, i.e. $\mathbf{W}$ and $\mathbf{b}$, the autoencoder will learn a representation of the input that is compressed. This can also be seen as a dimensionality reduction. This enables the discovery of latent stuctures in high dimensional spaces. Imposing a sparse representation can also be seen as limiting the "activation" of the hidden units, using for instance the Kullback-Leibler (KL) divergence. By using the average activation of the hidden layer, sparsity can be enforced. A sparsity parameter is seleted, denoted p, which is a positive number close to zero. The KL divergence between the average activation and the sparsity parameter is then minimized using the following:

$$J_{KL}(p||\hat{\mathbf{p}}) = \sum_{j=1}^{n} p \log \frac{p}{\hat{p_j}} + (1-p) \log \frac{1-p}{1-\hat{p}_j} \tag{7}$$

Here $\hat{\mathbf{p}}$ is a vector of average activations, i.e. per hidden-layer. This combined with a weight decay and the reconstruction error provides the cost function used for a sparse autoencoder, [4]

$$J(\mathbf{W}, \mathbf{b}) = J_E(\mathbf{W}, \mathbf{b}) + \beta J_{KL}(p||\hat{\mathbf{p}}) + \frac{\lambda}{2} \sum_{l=1}^{2} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (w_{ij}^l)^2 \tag{8}$$

Here β regulate the sparsity penality, and λ is a standard weight decay term while s_l and s_{l+1} is the size of connected hidden layers.

3. NON-NEGATIVITY CONSTRAINTS

Much like NMF, the idea of part-based representation, intuitively consists of combining in an additive way. By applying non-negativity constraints on a sparse autoencoder, the reconstruction of the input data is claimed by [4] to be improved, since the autoencoder is capable of decomposing data into sparse elements and additively combine them at the decoder. The work in [4] proposes non-negativity by replacing the weight decay in Eq. (8). This replaces λ with α which is greater or equal to zero. Moreover the function $f(x)$ is introduced on the weights which applies a non-negativity constraint, regularized by α.

$$f(w_{ij}) = \begin{cases} w_{ij}^2 & w_{ij} < 0 \\ 0 & w_{ij} > \end{cases} \tag{9}$$

Updating the weights and the biases is done with the use of backpropagation algorithm.

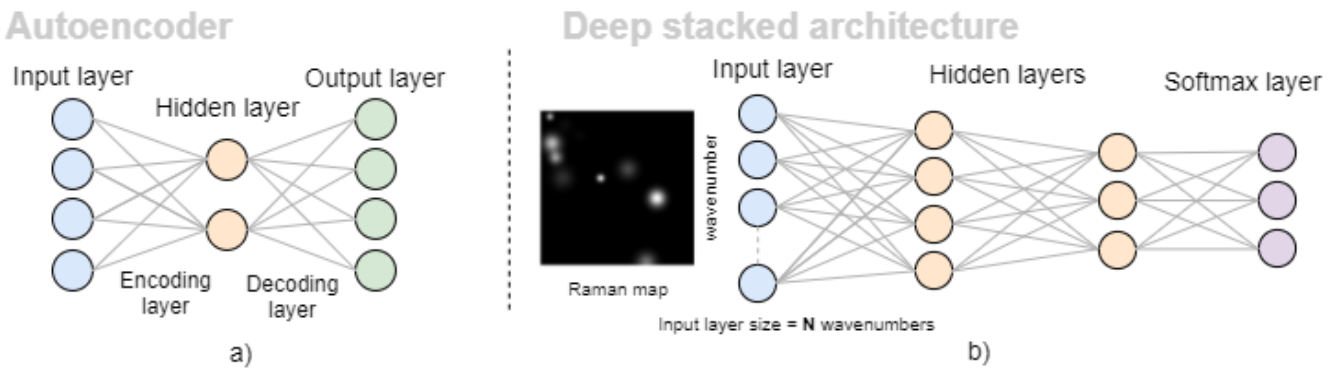

Fig. 1: a) shows a regular autoencoder consisting of an encoding and decoding layer. b) shows the deep stacked architecture with nonnegative constraints used for this work. Spectrum of each Raman map, consisting of 1000 wavenumbers is given as input.

4. SETUP

As proposed in [4] and [8], a layer-wise training is used to pre-train each layer unsupervised such that features are learned. In this work, the deep nonnegative constrained autoencoder (NCAE) is replicated using the layered training approach. This means each layer of the deep architecture is trained individually, using a stacked approach. In other words, the hidden activations of the previous trained autoencoder is used as input for the next. The final autoencoder is used as input to a softmax classifier. After the individual stacked training approach, the whole network is combined and trained in a supervised fashion, e.g. finetuned. The authors propose the use of a batch gradient descent algorithm called Limited-memory BFGS (L-BFGS) quasi-Newton method which approximates the hessian matrix. However, since the implementation was done in TensorFlow (TF) a native TF optimizer, e.g. Adam, was used instead.

4.1. Verification

For checking that indeed the deep-learning model, NCAE, is implemented correctly, a verification using the MNIST data set has been produced. The parameters used in the verification are the same as in [4] with the exception of the nonnegativity constraint penalty, α which is changed. For illustration, the used parameters are displayed in Table 1,

Parameters	NCAE
Sparsity penalty (β)	3
Sparsity parameter (p)	0.05
Weight decay penalty (λ)	-
Nonnegativity constraint penalty (α)	0.1
Convergence tolerance	1e-9
Maximum No. of Iterations	400

Table 1: NCAE parameters used for verification using MNIST data

The latent space after finetuning is displayed in Fig. 2. As it can be observed, the handwritten digits are better separated in the second layer.

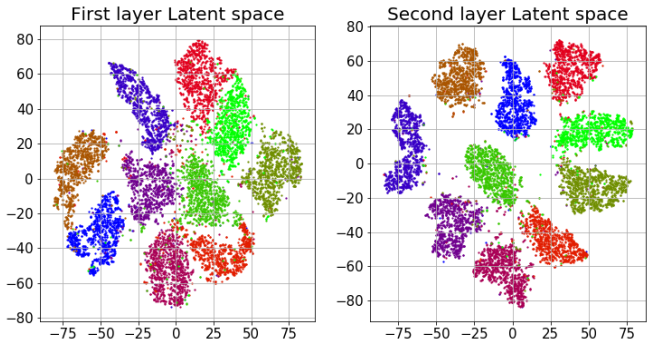

Fig. 2: MNIST latent space after finetuning

We also created a confusion matrix, Fig. 3. The classification percentages are very high (greater and equal to 95%) for both the validation and test datasets. This indicates that indeed the model works as hoped.

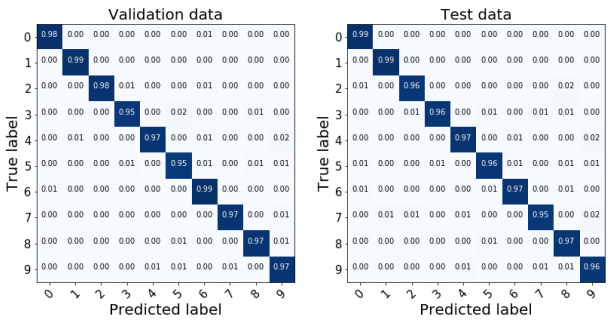

Fig. 3: MNIST confusion matrix

4.2. Simulated Raman data

A simulated SERS dataset is created using MATLAB. The dataset is created using parameters such as the number of wavenumbers, and the size of the Raman map with additional parameters such as; a) Signal-to-background-ratio (SBR) that determines the intensity of the peaks with respect to the background. b) K the number of Voight-profiles, e.g. the number of "substancens" or components in the spectrum and c) the number of hotspots in each Raman map.

The dataset used for training consisted of 1000 wavenum-

bers using a Raman map size of 100 × 100 samples. The number of hotspots was 10 with an SBR of 0 dB. Measurement noise was given as 10^{-3}. Two different substances were created using the same parameters, however varying K. The resulting training data, Raman spectra, consisting of $K = 1$ and $K = 2$ can be observed in Fig. 5 on the top most subplots. These are for the remainder of the paper referenced as Component 1 and Component 2.

The test data is a mixture of both components with an added "anomaly" to the spectrum. This is to emulate a nonlinearity in the spectra when mixing components. The mixture levels are given in steps of 10%, e.g. one test set consists of 10% of Component 1 with 90% of Component 2, and so on.

The simulated hotspots can be visualized in Fig. 4.

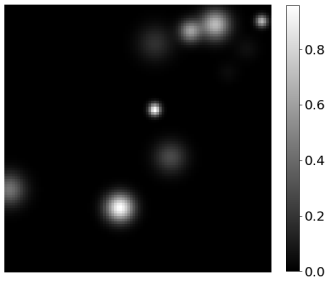

Fig. 4: Hot spots

Also, for visualization purposes, the profiles of the two "substances", class 0 and class 1, and the background signal, class 2, are displayed in Fig. 5.

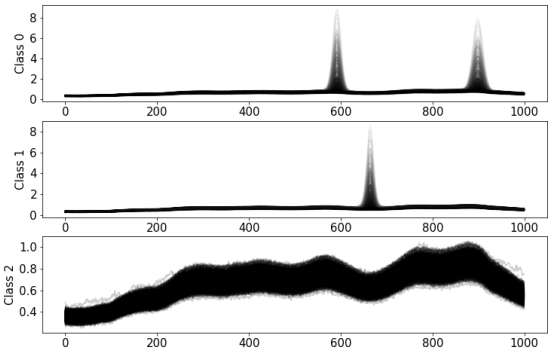

Fig. 5: Raman simulation data

5. RESULTS

This section contains the main results of the deep learning model described in the Setup Section 4 for the simulated raman data set.

As described, the model is a classification model with three classes with class 2 used for collecting the background signal.

In order to check that indeed the model works as expected, several testing figures have been generated. These figures are presented below with a small description.

The first indication that the model functions correctly is the component identification. In terms of peak and wave numbers the components in Fig. 6 look very similar to the ones described in the Setup Section 4, Fig. 5.

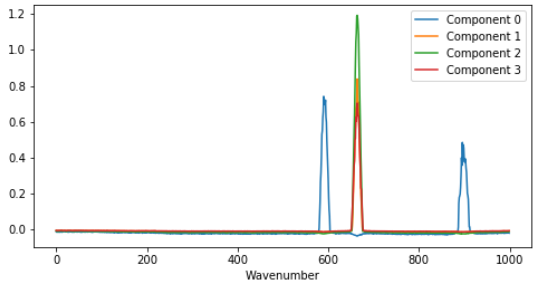

Fig. 6: Component identification

The weights for the second layer and softmax layer are visualized in Fig. 7. From Fig. 6 we see that units 1 and 2 are related to class 1. Further we see that these are related to components 3 and 1 + 2. Comparing Fig. 5 to Fig. 4 it is easily seen that this is class 1. The same can be done for class 0. In Figure 7 we also see that activations go down for units 1 + 2 as the concentration of class 0 goes up and vice versa for unit 0.

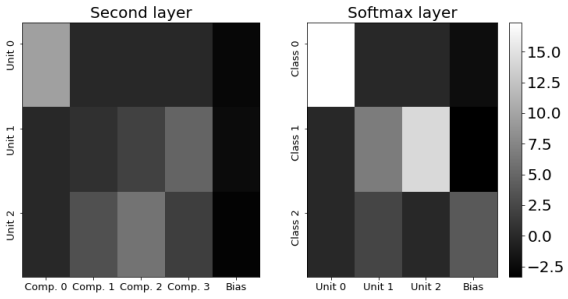

Fig. 7: Weights for the second and softmax layers

An overall overview of the activations and probabilities over the D space can be viewed in Fig. 8. As it can be observed, the hot spots are activated (lighter tones) in both images similarly to the original hot spot simulation image, Fig. 4. For probabilities, the values are slightly biased towards class 1 (even though the concentration of class 0 is 0.9, it is still concluding that class 1 is present in the center of the hot spots).

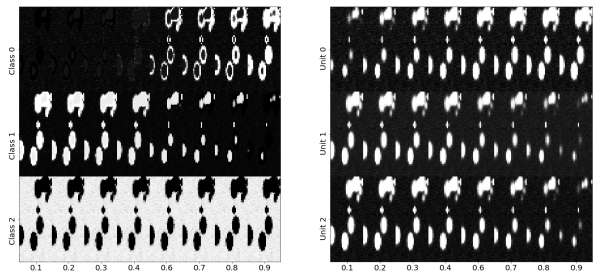

Fig. 8: Left is the probabilities of each class, to the right is the activation of each hidden unit when changing the concentration of each substrate.

We also plotted a running average to check that indeed the probabilities for class 0, 1 and 2 (blue, green and purple solid lines) over space D behave as hoped. As one can see in Fig. 9, the behaviour is consistent with Fig. 8.

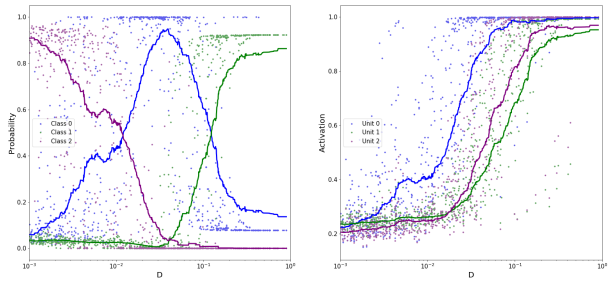

Fig. 9: Probability versus D space for a concentration of 0.7 for class 0 and 0.3 for class 1. Lines are running averages with a window size of 100.

We plotted the confusion matrices for the test and training data, Fig. 10. When investigating the miss classifications, we find that when $D > 6 \cdot 10^{-3}$, the model is predicting correctly. This indicates another tuning parameter should be used in terms of the threshold defining the background described in the Setup Section 4.

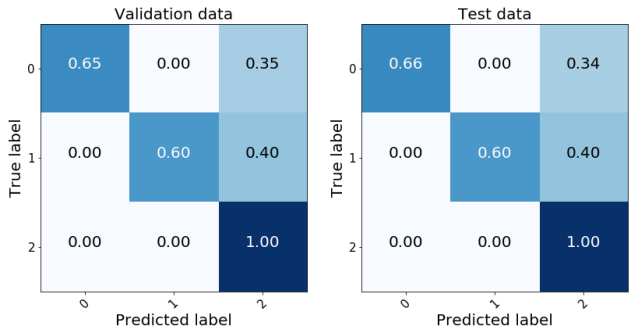

Fig. 10: Confuson matrix for the raman data

In order to check how the model deals with the concentrations, we also made a predicted concentration using a simple transformation of the activations, Fig. 11. As the figure shows, the model performs quite poorly since the intersection should be close to the gray diagonals.

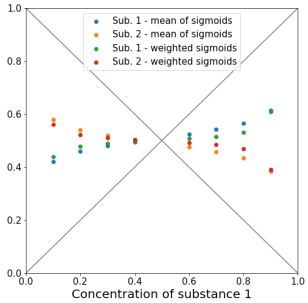

Fig. 11: Predicted concentration using activations from sigmoid function

Finally, we also generated a comparison between NMF and the deep learning model for predicting concentrations, Fig. 12. As it can be seen in the figures, NMF performs much better than the deep learning model. The difference between these results is described into depth in the Discussion Section 6.

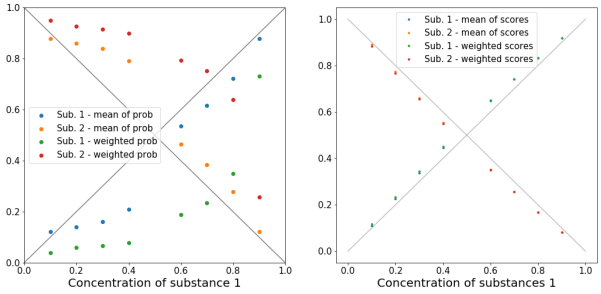

Fig. 12: Predicted concentration using probabilities (left) and NMF (right)

6. DISCUSSION

The estimated components shown in Fig. 6 are essential peak detections, which is sufficient for a classification task. It is clear that the estimated components are not strictly non-negative. This is due to the non-negative formulation (3) being a soft-constraint, which is in contrast to the usual hard-constraint (projection) formulation used in NMF. A way to improve this is to use the same update strategy as Seung and Lee [5]. Besides achieving strict non-negativity this also has the advantage that we decrease the number of hyper parameters that has to be tuned.

Intuitive a 3 component NMF would be able to reconstruct the the two signals in our data (one extra component is needed for the background). When using NCAE we were unable to achieve a satisfying first layer using only 3 components and needed to use 4 instead. As the background doesn't have to be modelled with the purpose of classification, this mean that we

end up with 3 components detecting the same peak. One solution to this would be to drop redundant components when fine tuning the final network (or even when increasing the depth of the network).

It is clear that the method to achieve a sparse solution (2.2) has its drawbacks, as one way to satisfy this, is to decrease the maximum activation during training. In our experience, this is exactly what happens, at least when using a too small batch size. However, increasing the batch size too we loose the benefits of this strategy. Furthermore, when applying the method to spectroscopic data, this formulation changes interpretation when moving from pre-training to fine tuning the full network. The reason for this is, as mentioned above, the background component is no longer needed, hence a sparser solution should be searched for. An alternative to (2.2) would be to apply a sparse prior to the estimated components as done in [9, 10].

The resulting deep neural network is an effective classifier when detecting when there is a signal, with apparent issue illustrated in Figure 10 being mainly an issue of tuning our threshold for creating the background class. When the network is applied to mixed data, an undesired behaviour is observed in both Figure 8 (left) and 12 (left). The probabilities for classes 0 and 1 losses their interpretation, as the Softmax function enhances even small differences in its inputs. However, in Figures 8 (right) and 9 (right) we see that the output of the sigmoid activation functions is a non decreasing function of the amount signal (both when varying D and concentration). From Figure 11 and 12 (right) it is clear that a extra layer is needed in order to perform on the same level as NMF. This could be done by changing the Softmax layer into a regression layer.

7. CONCLUSION

We applied NNCAE to simulated SERS data consisting of pure signals to pre-train a deep neural network with a Softmax layer on top. The resulting network was with high precision able to distinguish pure signals from each other. When we applied the network to mixed data, the output probabilities from the Softmax layer wasn't related to the actual concentrations. The sigmoid activation functions indicated that changing the last layer into a regression layer, the network would perform comparable to NMF.

8. REFERENCES

[1] Hong Wei and Hongxing Xu, "Hot spots in different metal nanostructures for plasmon-enhanced Raman spectroscopy," *Nanoscale*, vol. 5, no. 22, pp. 10794, oct 2013.

[2] Paul Sajda, Shuyan Du, and Lucas Parra, "Recovery of constituent spectra using non-negative matrix factorization," .

[3] Tommy S. Alstrom, Mikkel N. Schmidt, Tomas Rindzevicius, Anja Boisen, and Jan Larsen, "A pseudo-Voigt component model for high-resolution recovery of constituent spectra in Raman spectroscopy," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. mar 2017, pp. 2317–2321, IEEE.

[4] Ehsan Hosseini-Asl, Jacek M. Zurada, and Olfa Nasraoui, "Deep Learning of Part-Based Representation of Data Using Sparse Autoencoders With Nonnegativity Constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 12, pp. 2486–2498, dec 2016.

[5] H. Sebastian Seung and Daniel D. Lee, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, oct 1999.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," 1977.

[7] Amy N. Langville, Carl D. Meyer, Russell Albright, James Cox, and David Duling, "Algorithms, Initializations, and Convergence for the Nonnegative Matrix Factorization," jul 2014.

[8] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," .

[9] Mikkel N. Schmidt and Rasmus K. Olsson, "Single-channel speech separation using sparse non-negative matrix," in *Interspeech 2006 and 9th International Conference on Spoken Language Processing, Interspeech 2006*, 2006, pp. 2614–2617.

[10] Tommy S. Alstrm, Kasper B. Frhling, Jan Larsen, Mikkel N. Schmidt, Michael Bache, Michael S. Schmidt, Mogens H. Jakobsen, and Anja Boisen, "Improving the robustness of surface enhanced raman spectroscopy based sensors by bayesian non-negative matrix factorization," in *2014 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 2124, 2014, REIMS, FRANCE*. 2014, IEEE.