Computer Science • 16 (2) 2015 http://dx.doi.org/10.7494/csci.2015.16.2.133

Mariusz Kleć
Danijel Koržinek

PRE-TRAINED DEEP NEURAL NETWORK USING SPARSE AUTOENCODERS AND SCATTERING WAVELET TRANSFORM FOR MUSICAL GENRE RECOGNITION

Abstract

Research described in this paper tries to combine the approach of Deep Neural Networks (DNN) with the novel audio features extracted using the Scattering Wavelet Transform (SWT) for classifying musical genres. The SWT uses a sequence of Wavelet Transforms to compute the modulation spectrum coefficients of multiple orders, which has already shown to be promising for this task. The DNN in this work uses pre-trained layers using Sparse Autoencoders (SAE). Data obtained from the Creative Commons website jamendo.com is used to boost the well-known GTZAN database, which is a standard benchmark for this task. The final classifier is tested using a 10-fold cross validation to achieve results similar to other state-of-the-art approaches.

Keywords Sparse Autoencoders, deep learning, genre recognition, Scattering Wavelet Transform

Citation Computer Science 16 (2) 2015: 133–144

1. Introduction

Genre recognition has been a staple of Music Information Retrieval (MIR) since the very beginning. Initial approaches relied mostly on Data Mining and Natural Language Processing, but audio analysis became popular when Machine Learning techniques improved to a substantial degree. MIR, as a concept, involves many diverse fields of study: classification, analysis, organization, recommendation, and various areas of research: signal processing, music theory, linguistics, sociology, psychology, and others. Many tasks that involve MIR will concentrate on a single problem by utilizing a particular method, but we are more often faced with projects that involve a variety of concepts spanning a couple of different domains.

If we take music recommendation as an example, it is clear that taking a single criterion into account most likely will not suffice to meet our goals, whether they be measured in terms of commercial success or user satisfaction. Problems like this have to be viewed from different angles and utilize several approaches to find a solution. The goal of finding the right music for the customer should consider not only the musical piece, but also the user and his needs. Additionally, both the music and the user have to be considered in context. The context of the music can be extracted from its acoustic features (genre, style, tempo, emotion, etc.) and meta-information (band, language, historic, social, etc.), while the user will have its own internal context (social, psychological, emotional), but also external context (environmental, situational). All of these can affect the quality of the system to a certain degree.

Even though genre recognition is not the best feature when it comes to MIR problems, it is still very popular among researchers. This comes as a consequence of its simplicity, both as a computational problem and a topic that is easy to understand by someone without a deep technical or music background. Everyone has heard of music genres, and it is very simple to construct the task as a classification problem with various types of inputs and a set of discrete classes as the output. In reality, genre recognition is a quite difficult and poorly defined problem. Not only is it difficult to assign a single class to any random musical piece, but even the classification taxonomy can not be defined without dissension. This has not stopped people from trying, and several standard benchmarks have been created to tackle this particular problem.

One of the most popular is the GTZAN [22] database, which is available for free and very easy to use. It consists of 1000 tracks (each 30 seconds in length) and organized into 10 classes (each consisting of 100 tracks). Initial experiments relied on simple musical descriptors (rhythm, pitch, timbre) as well as classic music analysis features like the Mel-Frequency Cepstral Coefficients (MFCC) and, less frequently, the Wavelet transform [8]. In [22], Tzanetakis utilized Gaussian Mixture Models on MFCCs, to achieve 61% baseline accuracy in the first ever GTZAN experiment. The authors in [19] reported 83% accuracy using Deep Neural Networks and spectral features. A breakthrough in feature quality was presented in the paper about the Scattering Wavelet Features (SWT) [1], where a simple SVM classifier achieved 89.3% accuracy. Later, the same features were utilized in a better Sparse Representation

Classifier [6], improving the result slightly with a reported accuracy of 91.2%. Other experiments on GTZAN utilized Wavelets [14] to achieve 78.5% accuracy, Deep-Belief Networks [9] for 84.3% accuracy, various representation based on the properties of the auditory cortex [17] for 92.4% accuracy and Compressive Sampling techniques [5] reporting 92.7% accuracy.

It is worth noting that the margin of error between these results is quite wide, and the difference at the high end becomes quite negligible due to the fairly small size of the corpus, compounded by the numerous reported inconsistencies within the database [21]. Ultimately, as mentioned in the previous paragraph, the genre taxonomy cannot be too objective, and individual sample track classification can often be fuzzy. As an example, many of the experiments mentioned above used voting to determine the final class; but, if the distribution of classes for individual frames gives, for example, 49% to one class and 51% to another, it may be difficult to say that either class is more relevant.

The goal of this paper is to combine the SWT described in [1] with the power of a Deep Neural Network (DNN) consisting of multiple layers of Sparse Autoencoders (SAE). To improve the Unsupervised Pre-training phase, a much larger database (acquired from the jamendo.com website) was prepared to match the GTZAN database. Jamendo is a music-sharing platform which publishes music on a Creative Commons license. A publicly-available API allowed the authors to download more than 80,000 musical tracks, nearly 10,000 of which were selected according to the GTZAN genres.

2. Background

This section includes background information of various components used in the experiments described in this paper.

2.1. Scattering Wavelet Transform

Most of the research behind MIR relies on Mel-Frequency Cepstral Coefficients (MFCCs), which are a Fourier-based feature set designed specifically for analyzing speech and music. MFCCs are calculated as the Fourier transform of the logarithm of the Fourier transform of the signal that was partitioned using standard windowing techniques (like in the STFT). The resulting features can be used to estimate a smoothed spectral envelope that is robust to small intra-class changes, but loses information [15].

Unlike the Fourier transform (which decomposes the signal into sinusoidal waves of infinite length), the Wavelet Transform (WT) encodes the exact location of the individual components. The Fourier transform encodes the same information as the phase component, but this is usually discarded in the standard MFCC feature set. This means that Fourier-based methods are very good at modeling harmonic signals, but are very weak at modeling sudden changes or short-term instabilities of the signal – something that the WT seems to deal with very well. The WT begins by defining a family of dilated signals known as wavelets. A single mother wavelet $\psi(t)$ is expanded to a dictionary of wavelets $\psi_{u,s}$, translated to u and scaled by s, using the formula:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right) \tag{1}$$

These wavelets are then used to decompose the input signal by using a convolution operator (denoted by $< \cdot >$):

$$Wf(u,s) =< f, \psi_{u,s} >= \int f(t)\frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right)dt \tag{2}$$

The Scattering Wavelet Transform (SWT) [15] works by computing a series of Wavelet decompositions iteratively (the output of one decomposition is decomposed again), producing a transformation which is both transformation invariant (like the MFCC) and experiences no information loss (proven by producing an inverse transform – something which cannot be done using MFCC without loss).

In [1], the SWT is used in the problem of phoneme classification and musical genre recognition. The paper also points out a similarity between the multilayer structure of the SWT and other deep structures, such as the Convolutional Neural Network [12]. This would hint at a certain level of redundancy of using DNNs with SWT features, but [6] demonstrates that certain improvements can still be achieved using better classifiers, and this paper intends to explore this.

2.2. Unsupervised feature learning

Training an Artificial Neural Network (ANN) with multiple layers (i.e., more than 2 or 3 hidden layers) using backpropagation does not fully utilize its theoretical capabilities. This is caused by the weakness of the gradient descent optimization method, where gradients that are computed by backpropagation rapidly diminish in magnitude as the depth of the network increases. As a result, the final layers don't receive meaningful training data [7]. This problem was well known and has been studied for decades. It was especially troubling that a Multi-Layer Perceptron often performed worse than its shallow counterparts (e.g., SVM) even though its expressiveness was theoretically more powerful.

A breakthrough happened in 2006 when G. E. Hinton introduced a fast-learning algorithm for training, which he named Deep Belief Networks [10]. This method uses a greedy layer-wise training to train one layer at a time in an unsupervised manner. This step is called pre-training, and its aim is to prepare the weights of the model in such a way that they better represent local feature states. Following this, the final fine-tuning of the weights using labeled data creates a model which performs far better than one that is trained on randomly-initialized weights alone.

This unsupervised pre-training approach started a new research trend called "deep learning." Deep learning takes advantage of unlabeled data to learn a good representation of the features space [2] – each layer representing another abstraction of the features pre-trained from a previous layer. Layer-wise, bottom-up pre-training (one layer at a time) is possible by incorporating Restrictive Boltzman Machines (RBM) or Autoencoders (AE) [3]. Stacking RBMs or AEs (as features detectors) forms a "deep structure" which can be fine-tuned using gradient-based optimization methods with respect to labeled data (i.e., supervised training).

2.3. Sparse Autoencoders

An Autoencoder (AE) is an ANN with an odd number of hidden layers, where the number of units in the output layer is set to be equal to the number of units in the input. In other words, AEs try to reconstruct the input at the output passing data through hidden layers. To ensure that the mapping is non-trivial, various constraints can be used to force the network to learn useful representations of the data. When the number of units in the hidden layer is smaller than the input, the AE learns a compressed form of the data, similar to the Principle Component Analysis (PCA). Unlike PCA, however, the learned compression is non-linear and more robust. If we use more hidden than input units, the AE can still learn meaningful representations of the data [3], provided it uses proper constraints.

One of the constraints that can be applied to AE training is trying to reconstruct the input from its corrupted version. This is the basic idea behind Denoising Autoencoders [23]. Another type of AE (used in this paper) is the Sparse Autoencoder (SA). [18, 13]. The idea behind it is to enforce activations of hidden units to be close to zero for most of the time during training. This can be achieved by applying the measure of Kullback-Liebler Divergence (KL) to the cost function:

$$KL = \rho \log \frac{\rho}{\hat{\rho}} + (1-\rho) \log \left(\frac{1-\rho}{1-\hat{\rho}} \right) \tag{3}$$

$$J_{sparse}(W, b) = J(W, b) + \beta \cdot KL(\rho || \hat{\rho}) \tag{4}$$

KL measures the difference between the two distributions: $\hat{\rho}$, which represents the average activations of hidden units over the training set, and ρ, which represents the target distribution. $J_{sparse}(W, b)$ denotes the sparse cost function with respect to weights W and biases b. Because we want to keep hidden units inactive most of the time, the target distribution should be set close to zero. In our experiments (described below), the target distribution ρ was always set to 0.1. In other words, we wanted to enforce $\hat{\rho} = \rho$. In order to penalize an average activation of hidden units which deviates too much from its target value of ρ, a special penalty term β is introduced to control the weight of the sparsity term.

2.4. DNN implementation

A neural network with mini-batch stochastic gradient descent (SGD) was developed in Matlab. The core of the code was written according to the guidelines presented in CS294A Lecture notes [16]. Additionally, the part of the code responsible for gradient calculation is compatible with the minFunc function that uses the L-BFGS [20] optimization algorithm. This algorithm uses a limited amount of computer memory, and was used in this paper for training the Autoencoders to improve training speed. The code, besides having an implemented square-error cost function, was extended to operate on cross entropy error [4] and to use momentum. The regularization term of $\|W\|^2$ was added to the cost error function, for the purpose of decreasing the magnitude of the weights and help to prevent overfitting. As a weight initialization for training AEs and NNs (in the case of experimenting without pre-training phase), we used a random uniform distribution U from the range described by formula 5, as it is recommended in [7]. The $n_{visible}$ and n_{hidden} denote the number of visible and hidden units in a given layer.

$$W_{init} = U\left[-\sqrt{\frac{6}{n_{visible} + n_{hidden}}}, \sqrt{\frac{6}{n_{visible} + n_{hidden}}}\right] \tag{5}$$

3. Data preparation

Two databases were used in the experiments. First is the well-known GTZAN dataset [22], consisting of 1000 musical files that are each 30 seconds long. They are categorized into 10 genres with 100 musical pieces per category (rock, blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae). The second data collection was obtained from the jamendo.com website, which offers music ready to download for free due to the Creative Commons license. A publicly-available API allowed us to download over 80,000 musical tracks, together with meta-data in an XML format. The meta-data contains, among other features, a genre association of each file. There are three attributes containing this information: "album genre", "track genre", and "tags". The "album genre" and "track genre" contain ID3 genre names, and "tags" can contain genres and other information (without restrictions) as annotated by users.

The goal was to create a much bigger database than GTZAN yet organized in the same manner. From the 80,000 files, only those that belonged to one of the 10 musical genres were taken into consideration. To avoid ambiguities, all of the files were passed through a couple of filters. Initially, files that had the same values in all attributes were immediately accepted. This assumption gave the highest probability that a particular file belonged to the given genre. For the genres that thusly resulted in less than 1000 musical files (this occurred with blues, country and reggae which are more specific than pop or rock), the filter was made less restrictive. First, only "track genre" and "album genre" had to be equal to choose a song (ignoring the tags); if there were still too few songs, only "track genre" was considered, ignoring the rest of the attributes. This generated a list of 9966 musical files organized into 10 musical genres with nearly 1000 track per genre.

Out of each file, a 30-second fragment starting at 30 seconds from the beginning of the file (to skip the potential problems which occur in the beginnings of some tracks) was extracted and down-sampled to 22,050 Hz (to match the GTZAN format).

The features were extracted from the files using the ScatNet toolbox. The SWT transform was computed to the depth of 2, as this was shown as the optimal setting in [1]. The first layer contained 8 wavelets per octave of the Gabor kind, and the second had 2 wavelets per octave of the Morlet type. The window length was set to 740 ms. After the transformation, we obtained 81,052 training examples from GTZAN and 802,925 training examples from the JAMENDO database – each with 747 features. The resulting databases can be acquired by contacting the authors.

4. Experiments

One of the goals of the experiments was to determine if the JAMENDO database could be used as an additional source of data for pre-training the DNN. We assumed that this data was completely independent from that in GTZAN, which was used for fine-tuning. In the first step, each SAE was trained using the songs from JAMENDO. The SAEs were trained with the L-BFGS optimizer for 300 epochs. When the training of the first SAE finished, the new data representation was derived by feeding the original data through the SAE's hidden layer. This representation was used for pre-training the second SAE and so on, until the whole DNN was pre-trained. This process of NN pre-training is illustrated in Figure 1.

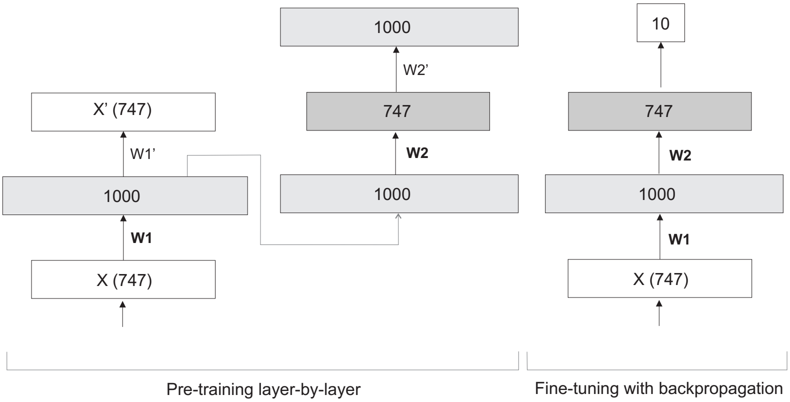

Figure 1. The process of pre-training the two hidden layers is illustrated. Two SAEs are trained. The weights from the encoder parts (W1 and W2) are used to initialize the final NN. Finally, the whole structure is fine-tuned using backpropagation, with the cross-entropy cost function.

To estimate the strength of the sparsity constraint β for the SAE, logistic regression was trained on the SAE representation derived from the GTZAN. The highest accuracy in this test determined the parameter β for the final SAE training. In each case, the target distribution of hidden activation ρ was set to 0.1.

Our experiments were based on pre-training and fine-tuning different topologies of neural networks. The GTZAN songs were randomly shuffled and divided into 10 folds for cross-validation (CV) tests. During CV, one fold was always reserved for validation and didn't take part in training. Its error rate was monitored during training to determine the early stopping criterion. The training was terminated when the cost value on the validation set didn't decrease by more than $1e-4$.

Before training, the data was standardized to achieve zero mean and a standard deviation of 1. The mean and standard deviation were calculated once in each fold and then used to standardize a test and validation set. Maximum voting was used to predict the label (genre) of the whole track in the test set. Classification error rates were averaged over all 10 folds. The final DNN had a topology consisting of 1,000, 747, 625, and 1,000 units in individual hidden layers respectively (see Table 1). The input vector had 747 dimensions. A log-sigmoid transfer functions were used in the DNNs and SAEs.

Table 1

Results after performing 10 fold CV on different topologies of DNNs pre-trained using SAEs. The results were determined by early-stopping. The experiment with the asterisk used momentum and a larger batch size.

Topology	Error %
747/1000/747/625/1000/10	11.1
747/1000/747/625/1000/10*	10.8
747/1000/747/625/10	10.9
747/1000/747/10	9.8
747/1000/10	12.1

Some additional experiments were also performed. A single fold of data was trained through 200 epochs. We plotted the changes of cost values for different topologies of NNs. Figure 2 presents the NNs with no pre-training whilst Figure 3 shows NNs with pre-training using SAEs. The experiments were performed with the following settings in both cases: learning rate: 3e−2; batch size: 80, momentum: 0.5, regularization: 1e−4.

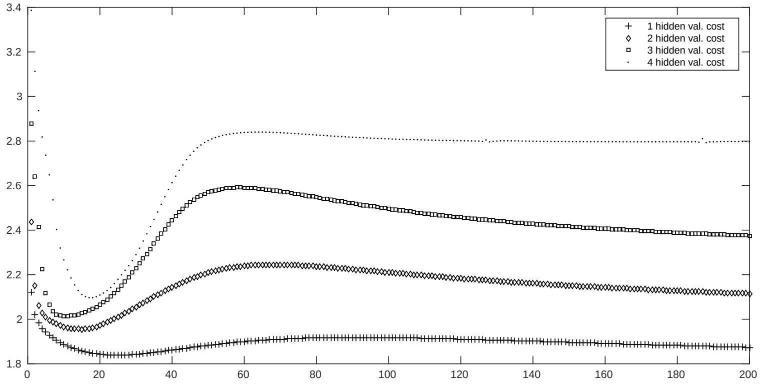

Figure 2. Validation set cost value of the network without pre-training on one fold of data.

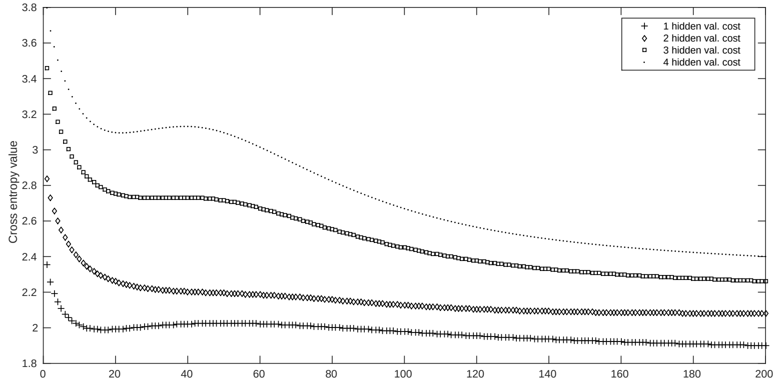

Figure 3. Validation set cost value of the network with pre-training on one fold of data.

Experiments were performed on a computer with 8 CPU threads (Intel i7 3820 3.6 Ghz) and also on an NVidia GeForce GTX TITAN Black GPU. The experiments used about 3 GBs of memory. Moving the calculation on the GPU improved the calculation slightly, but further code optimization is required for more significant improvements.

5. Conclusions and discussion

In our previous paper, we showed that adding more layers to the MLP does not improve the accuracy of genre recognition, and may even diminish it if the number of parameters becomes too high [11]. Using a hidden layer that was pre-trained with an SAE did improve the accuracy, however.

The purpose of the experiments in this paper was to examine whether pre-training using SAE improved the genre recognition in more than one hidden layer. This is the basic principle for building a DNN that has been proven to work for many tasks, including genre recognition [9]. The difference in our work is the utilization of SWT, which already outperforms many of the other approaches, including the DBN mentioned earlier.

The fine-tuning by using gradient descent didn't always improve the final network error rate. The best result was obtained with two hidden layers (9.8% error rate), but we weren't able to reproduce this improvement for other topologies with higher number of layers.

The graphs in Figures 2 and 3 demonstrate this problem very well. The network that didn't use pre-training achieved its minimum cost very early in the training (around 20–30 epochs) and didn't improved that score after that. It seems to over-fit quickly and converges to a worse value than achieved in 20–30 epochs. The shape of the cost for the network with pre-trained weights, however, has a much different shape. Not only is the over-fitting much less pronounced, but the network seems to generally improve much better than its randomly-initialized counterpart.

One of the reasons for our results may be the early stopping strategy that we employed in our experiments. The problem with having such a small corpus is that small differences in training can cause large jumps in the test error rate, and the error rate is poorly correlated with the network loss. Some initial experiments showed that training the network for much longer than the early stopping suggested could improve the error rate significantly, but we are not sure about the objectivity of such a result.

Nevertheless, even if the end result could be improved in individual layers, it seems that when comparing the results between layers, adding more layers to the DNN simply doesn't improve either the network cost or the error rate in any of the training, validation, or test sets. It is not clear whether this is the consequence of using the SWT features or an issue with the training methodology.

More tests are planned for this problem, especially with respect to the early stopping issue mentioned above. Different methods of pre-training also need to be tested; for example, de-noising AE and RBMs. Finally, attempts at studying the feature-space in a spatio-temporal manner could enable completely different approaches to this problem. The current system models the problem in the feature-space of individual frames describing the spectral content of the sound at a certain point in time, completely disregarding the temporal aspects of the signal (i.e the change of frequencies in time). It is likely that analysing several samples at once will allow the system to recognize certain temporal patterns in the signal. Furthermore, such methods as Convolutional Neural Networks have shown very promising in analysing 2-D signals and would be worth investigating here as well.

Acknowledgements

We would like to thank prof. Krzysztof Marasek, Thomas Kemp, and Christian Scheible for their support. This work was funded by a grant agreement no. ST/MN/MUL/2013 at the Polish-Japanese Academy of Information Technology.

References

[1] Andén J., Mallat S.: *Deep Scattering Spectrum. CoRR*, vol. abs/1304.6763, 2013, `http://arxiv.org/abs/1304.6763`.

[2] Bengio Y.: Learning Deep Architectures for AI. *Foundations Trends Machine Learning*, vol. 2(1), pp. 1–127, `http://dx.doi.org/10.1561/2200000006`.

[3] Bengio Y., Lamblin P., Popovici D., Larochelle H., et al.: Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, vol. 19, p. 153, 2007.

[4] Bishop C. M.: *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., New York, NY, USA, 1995.

[5] Chang K. K., Jang J. S. R., Iliopoulos C. S.: Music Genre Classification via Compressive Sampling. In: *ISMIR*, pp. 387–392, 2010.

[6] Chen X., Ramadge P. J.: Music genre classification using multiscale scattering and sparse representations. In: *Information Sciences and Systems (CISS), 2013 47th Annual Conference on*, pp. 1–6, IEEE, 2013.

[7] Glorot X., Bengio Y.: Understanding the difficulty of training deep feedforward neural networks. In: *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[8] Grimaldi M., Cunningham P., Kokaram A.: A wavelet packet representation of audio signals for music genre classification using different ensemble and feature selection techniques. In: *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pp. 102–108, ACM, 2003.

[9] Hamel P., Eck D.: Learning Features from Music Audio with Deep Belief Networks. In: *ISMIR*, pp. 339–344, Utrecht, The Netherlands, 2010.

[10] Hinton G., Osindero S., Teh Y. W.: A fast learning algorithm for deep belief nets. *Neural Computation*, vol. 18(7), pp. 1527–1554, 2006.

[11] Kleć M., Koržinek D.: Unsupervised Feature Pre-training of the Scattering Wavelet Transform for Musical Genre Recognition. *Procedia Technology*, vol. 18, pp. 133–139, 2014.

[12] LeCun Y., Bengio Y.: The Handbook of Brain Theory and Neural Networks. chap. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258, MIT Press, Cambridge, MA, USA, 1998, `http://dl.acm.org/citation.cfm?id=303568.303704`.

[13] Lee H., Ekanadham C., Ng A.Y.: Sparse deep belief net model for visual area V2. In: *Advances in neural information processing systems*, pp. 873–880, MIT Press, 2008.

[14] Li T., Ogihara M., Li Q.: A comparative study on content-based music genre classification. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 282–289, ACM, 2003.

[15] Mallat S.: Group invariant scattering. *Communications on Pure and Applied Mathematics*, vol. 65(10), pp. 1331–1398, 2012.

[16] Ng A.: Sparse autoencoder. *CS294A Lecture Notes*, vol. 72, pp. 1–19, 2011.

[17] Panagakis Y., Kotropoulos C., Arce G. R.: Music Genre Classification Using Locality Preserving Non-Negative Tensor Factorization and Sparse Representations. In: *ISMIR*, pp. 249–254, 2009.

[18] Poultney C., Chopra S., Cun Y.L., et al.: Efficient learning of sparse representations with an energy-based model. In: *Advances in neural information processing systems*, pp. 1137–1144, 2006.

[19] Sigtia S., Dixon S.: Improved music feature learning with deep neural networks. In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 6959–6963, IEEE, 2014.

[20] Skajaa A.: Limited memory BFGS for nonsmooth optimization. Master's thesis, Courant Institute of Mathematical Science, New York University, 2010.

[21] Sturm B.L.: The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461*, 2013.

[22] Tzanetakis G., Cook P.: Musical genre classification of audio signals. *Speech and Audio Processing, IEEE transactions on*, vol. 10(5), pp. 293–302, 2002.

[23] Vincent P., Larochelle H., Lajoie I., Bengio Y., Manzagol P. A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

Affiliations

Mariusz Kleć
Polish-Japanese Academy of Information Technology, Warsaw, Poland, mklec@pjwstk.edu.pl

Danijel Koržinek
Polish-Japanese Academy of Information Technology, Warsaw, Poland,
danijel@pjwstk.edu.pl

Received: 19.01.2015
Revised: 09.03.2015
Accepted: 17.03.2015