An Autoencoder as a Naive Approach to Audio Source Separation

Erich Paul Andrag, 1355800

Supervisor: Dr Ben Graham

M2 Project Report in partial completion of the Erasmus Mundus Complex System Science Master's Program

***Abstract*—The separation of music signals into meaningful constituents is investigated through the use of an autoencoder. An autoencoder is an unsupervised artificial neural network which has been used with success in the processing of images. Here we apply the method to the time-frequency representation of a music signal with the intention that the method will be able to identify different source components of the signal. First we see if the autoencoder is able to recreate the input, this is the method by which an autoencoder is trained. The result shows that the network has difficulty recreating the input time-frequency representation, especially along the time axis. Secondly we look at the code layer, where the network provides an abstract representation of the inputs. Again the result shows that the time frame overpowers the activation of hidden units and no clear separation of the sources are observed. Significant explanation of possible improvements and alterations to the method is discussed.**

***Index Terms*—Autoencoder, Audio Source Separation, Time-Frequency Representation**

I. INTRODUCTION

SOURCE separation involves processing an input signal to reconstruct or separate the signal into its constituents. These constituents are required to be of interest to the end representation. As example, the input signal could be audio produced by a orchestra. A meaningful constituent would be the signal produced by one of the instruments contributing to the input signal, as opposed to just one note played by one of the instruments. This research is focused on audio signals produced by musical instruments.

The problem has gained some interest over the years and while techniques for music separation exists [1], [2], the separated sources contain considerable artifacts. The algorithms often also require to be trained for specific sources. Speech processing and separation, a closely related field of research, is more active, [2]–[8]. This is due to the importance of, among others, natural language processing, denoising, and the cocktail party problem. These challenges are all relevant in the development of smart devices. The difference between music and speech signals might be considered trivial, but this is not the case. Speech signals processed by a low pass filter is still recognizable and phase information in a speech signal is not necessary for the processing thereof [9], contrary to the processing of musical signals.

A formalization of the source separation problem can be found in [6] and more concisely repeated in [9]. The signal emitted by the j-source, $(1 \leq j \leq n)$, and $x_i(t)$ the signal recorded by the i-th microphone $(1 \leq i \leq m)$ is related through the filter $a_{ij}(\tau)$. We then have $x_i(t) = \sum_{j=1}^{n} \sum_{\tau=0}^{\infty} a_{ij}(\tau)s_j(t-\tau) + n_i(t)$ where $n_i(t)$ noise in the process. In our case we assume there to be only one microphone (signal) and thus i is assumed to be unity. To be consistent with other representations of source separation we reformulate. The input signal $y(t)$ is a combination of sources $x_i(t)$, $(1 \leq i \leq n)$ with some constant c_i capturing the relation. Final representation is then $y(t) = c_i x_i(t)$, where noise is modeled as being one of the sources $x_i(t)$ and although the constant c_i could vary over time, it is often assumed the mixing process has constant gain for each source.

The human hearing system is unequaled in its ability to discern and separate complex sounds into its respective sources. For example, listing to audio we could each recognize distinct source contributions and reproduce or imitate the source with varying levels of skill. This ability is also very apparent when we find ourselves in noisy environments, where we are able to identify and listen to a specific audio source.

For machines it is not as simple. Various algorithms have been employed to solve this problem with varying levels of success. What makes this problem difficult is that it is mathematically ill-posed [1]. There is not enough information in a signal to simply separate the sources, additional information has to be incorporated. This, typically, requires that the algorithm has prior information about the input signal. Blind source separation (BSS) in general refers to a case with no prior information about the input signal. The algorithm then has to discern what type of signal it is receiving. A further specialization is blind audio source separation (BASS). Although our research does not assume any prior information about the input or sources, we do train it on music signals. The algorithm thus does not act completely blind. Once the algorithm achieves satisfactory development it should generalize to separating blind sources.

Artificial neural networks have long been considered an imitation, of the processes in the brain, albeit a lacking imitation [10]. Further, with the recent and continuous advances in machine learning [11], [12], a simple Deep Neural Network (DNN) could be capable of separating and reconstructing individual sources, even if only to an elementary extent.

The idea in this research is then to determine if an artificial neural network could function as a source separation algorithm by learning the capabilities required to separate musical sources. This problem differs from previous uses of artificial neural networks in source separation. Firstly, the input signal is not a speech signal [13]. Secondly, the network is not simply used as a classification technique [14]. The network is required to learn a complex representation of a musical instrument and be able to identify this instrument among many others in the input signal. Specifically, a network type called an autoencoder will be used. An autoencoder, first introduced by the PDP group [15], has developed considerably through the "deep architecture" approach popularized by [11]. We will discuss the workings of this network in relevance to the research problem.

Our approach is so-called *naive,* since an autoencoder is employed without alterations to the standard functions and training process of the network. It is assumed that the time-frequency (TF) representation of the input signal can act as an image input to the network. An autoencoder has been shown to perform well on image classification and feature extraction [16]. This naive approach has the advantage of being very broad, although, as we will see some obvious pitfalls arise.

II. EXISTING APPROACHES

For the purposes of discussing and grouping the most common techniques used in audio source separation, we ignore the differences between algorithms adapted for either speech or music processing. As is the case in most of these frequently used algorithms, they can be readily adapted or fine-tuned to work on either or both types of input.

In general the techniques are classified as Gaussian Mixture Model (GMM) [1], [17], Hidden Markov Model (HMM) [8], Deep Neural Network (DNN) [4], [7], [13], [14], [18], Independent Component Analysis (ICA) [19], sparse decomposition [20], Non-negative Matrix Factorization (NMF) [2], [9], [21], or Computational Auditory Scene Analysis (CASA) [3], [22]. It is common for these techniques to be used in conjunction with each other. For example, NMF can be used as initialization of the algorithm while DNN is used to fine-tune the weights and classify the sources, as in [14].

Some of the techniques have become less popular. ICA is now rarely used, but was one of the standard algorithms for source separation tasks. CASA, which models the way the human hearing system works, is more often used as a support algorithm. Although more suited to speech processing, it models the spatial origin of sounds. Sparse decomposition forms a part of almost every technique and is rarely employed as a standalone algorithm.

The most common algorithm is NMF. The algorithm is used in almost all methods in some way or another and deserves special mention. NMF factorizes any non-negative matrix V into a basis matrix B, called a *dictionary,* and a gain matrix G, such that

$$V_i \approx B_i G_i, \tag{1}$$

where i is the sources $(1 \leq i \leq n)$. The dictionary matrix is trained for a specific source and stays constant, while the gain matrix is estimated when a input signal is fed to the algorithm. The factorization is done in the time domain, and further processing is often done in the TF representation.

The state-of-the-art method for music source separation is GMM, where the sources are modeled as sum of Gaussian random variables,

$$Y(t,f) = \sum_{j=1}^{J} X_j(t,f). \tag{2}$$

Here Y and X_j are complex valued vectors of the TF-representation, t is the time dimension and f the frequency dimension. The distribution of the sources $X_j(t,f) \sim N(0, \upsilon_j(t,f)\mathbf{R}_j(t,f))$ has zero mean and the variance depends on the spatial covariance matrix $\mathbf{R}_j(t,f)$ and non-negative spectral power scalar $\upsilon_j(t,f)$. The sources are further decomposed and finally the Expectation Maximization (EM) algorithm is used to find the most likely estimate of sources. This description of a typical GMM algorithm is adapted to our notation from [1] and is repeated here due to its success and detailed formulation. This method can work blind and even better with some information about the sources in the form of a dictionary. A demonstration can be found at [23].

The use of DNN for speech processing has increased rapidly in the last few years. With the advances in so-called "deep architecture" a Recurrent Neural Network (RNN), subclass of ANNs has managed to achieve one of the lowest classification errors on the TIMIT database [24]. The TIMIT database is a set of voice recordings from North American male and female voices and a general benchmark database for speech processing.

III. MODEL

There are two stages to the separation process. The first is to obtain a TF representation of the input signal and the second is to feed this representation to the separation algorithm. The stages act independently, but are very sensitive to one another. Care is taken to present the interface as informative as possible.

A. Time-frequency representation

Here we discuss the process of transforming an input signal into the TF domain. The Fourier transform is used for this purpose. A requirement of the transform is, if the sources are to be reconstructed after the separation process, the transform should be invertible.

So to begin, a discrete-time short-time Fourier transform (STFT) is produced on successive windows of the signal. To understand the process and its components better, mathematically it is written as

$$\text{STFT}\{y(t)\}(m,\omega) = \sum_{n=-\infty}^{\infty} y[n]w[n-m]\mathrm{e}^{-j\omega n} \tag{3}$$

where $w(t)$ is a window function, $y(t)$ the input signal, m the window size and n the time samples. In our case we us a Hamming window, commonly used in audio processing. Each STFT window produces a vector indicating the frequencies present in that time window. Several windows are concatenated to give a TF representation. The implementation of a STFT in practice is in the form of a fast Fourier transform (FFT). A FFT algorithm is generally accepted to be the fastest way to perform a Fourier Transform.

The advantage of a STFT comes from the process being invertible. There are several algorithms available to perform this inversion. In the case of audio processing, performing the STFT and its inverse results in a signal with very little discernible difference compared to the original input.

A further advantage of Fourier transforms are their linearity property. That is, linear operations in the time domain is equal to linear operations in the frequency domain. This is in fact a very important feature when determining the source separation problem.

$$y(t) = ax_1(t) + bx_2(t) \tag{4}$$

$$Y(t,f) = aX_i(t,f) + bX_2(t,f) \tag{5}$$

Here $y(t)$ is the time signal and $Y(t,f)$ is the STFT of the signal, f represents the frequency, a and b are constants.

B. Network Input

In general the TF representation is considered a more powerful input to source separation algorithms [1]. It typically produces sparse representation which, as input to an autoencoder, is very good [2].

Performing a Fourier transform on an input signal results in a complex values. The real part corresponds to the amplitude of the frequency and the imaginary part is the phase angle of the signal. This is problematic, since very few autoencoder networks have fully investigated the problem of complex-valued inputs [25]. Many algorithms overcome this problem by simply only working with the magnitude or power spectra [2], [9]. This is indeed the process we will follow, although it does make reconstruction of the source time representation more difficult, albeit not impossible. The phase for the separated sources can simply be assumed to be the same as that of the input signal [9] or more involved methods can be used as in [26]–[28] or several others. Continuing with only the magnitude, the equation in (5) then becomes

$$|Y(t,f)| = a\,|X_i(t,f)| + b\,|X_2(t,f)|\,. \tag{6}$$

Plotting $|Y(t,f)|$ results in what is called a spectrogram, and is used as the input to the next stage. A spectrogram is presented as a matrix with one axis indicating time and the other frequency. Each element in the matrix can be seen as similar to the pixel in a picture. The resolution of both axes are important as the resolution, or matrix size of a STFT instance is fixed. A trade-off then exist between the resolution of the axes, this trade-off parameter is called the bandwidth.

C. Autoencoder

Artificial neural networks (ANNs) involve multiplying the inputs $\bar{x_i}$ by a vector of values called weights $\bar{w_{i,j}}$ to produce outputs $\bar{o_j}$. Weights thus map input i to output j. Outputs are sent through an activation function, which allows the network to learn complex non-linear representations of the inputs. The outputs are further compared to the optimal outputs and an error is determined. The weights are then typically greedily adjusted to reduce the error of successive iterations. Development of ANNs has led to multilayered networks, where the output of the previous layer is used as the input to a following layer. This has increased their popularity to separate, classify, and learn complex problems. As mentioned previously, an autoencoder is a subclass of ANNs with multiple layers.

The autoencoder used in our source separation algorithm was developed by [11], [16]. An autoencoder involves two steps: firstly, to encode the input to an abstract representation and, secondly, to decode this abstract representation back to a reconstruction of the input. Hence the advantage of using an autoencoder is clearly illustrated, the autoencoder requires no classified inputs, i.e. it is trained in an unsupervised way. The quality of the reconstruction is its own measure of success. In our case we expect the abstract representation to indicate which sources are present.

Optimizing the weights in an autoencoder is difficult and most methods find poor minima [16]. To overcome this problem each layer is pre-trained using a restricted Boltzmann machine (RBM). RBMs take the inputs and then try and recreate the inputs through an intermediate hidden layer [29]. The inputs correspond to visible units, whereas the features is the outputs (or hidden layer) and correspond to hidden units. The process is depicted in fig. 1(a) and the same process applies to individually training subsequent layers of the autoencoder, fig. 1(b). Important here is that we vary the number of layers between experiments and the figures are simply for illustration of the workings of an autoencoder.

To create the complete autoencoder the individual RBMs are unrolled by stacking them successively. The resulting network is depicted in fig. 2(a). Here we can see the inputs are first encoded to produce a code layer. As discussed the code layer is the most abstract representation of the inputs. The code layer elements should correspond to features of different instruments. A set of features could together describe an instrument. A group of activated units in the code layer should indicate which instrument is contributing to the signal. Adding more layers to the autoencoder could result in capturing further higher-order correlations between the units in the layer below.

An important factor is the size of each layer, or rather the reduction in size between successive layers. Considering the case for reduction in size for each layer as the abstraction increases, this is analogous to compressing the representation of the input. Increasing the size of successive layers is not often done, since this provides for an over-complete representation and information becomes more sparse. In our case we keep the layer sizes similar, since the spectrogram input is already quite sparse and we do not want a compressed form of the input, rather a separation of the input.

The final network is fine-tuned using a standard feed forward network as in fig. 2(b). The inputs for the network is quite non-Gaussian so a good error function to use here is the cross-entropy error $[-\sum_i p_i \log \hat{p}_i - \sum_i (1-p_i)\log(1-\hat{p}_i)]$, where p_i is the intensity of pixel i and $\hat{p}_i$ is the intensity of pixel i in the reconstruction [16].

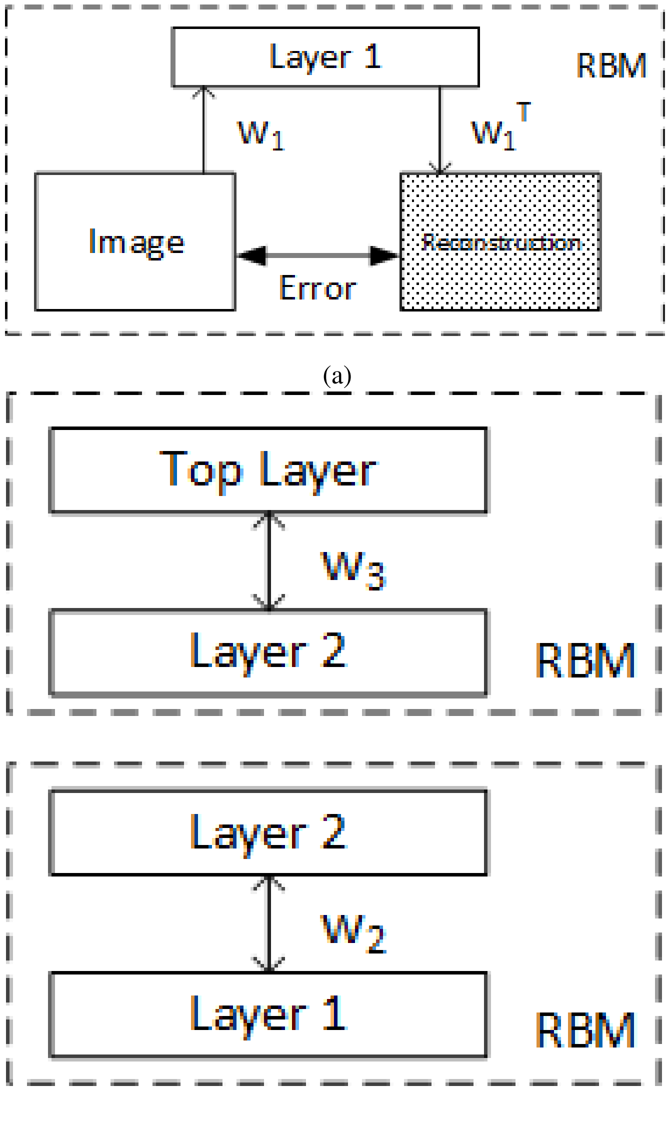


Fig. 1. The figure illustrates how an RBM functions. In (a) a detailed depiction of how the inputs are converted to the first layer and then multiplied by the transpose of the weights to create a reconstruction of the inputs. The error is measured and used to optimize the weights. In (b), successive separate layers are showed and they follow the same process as in (a).

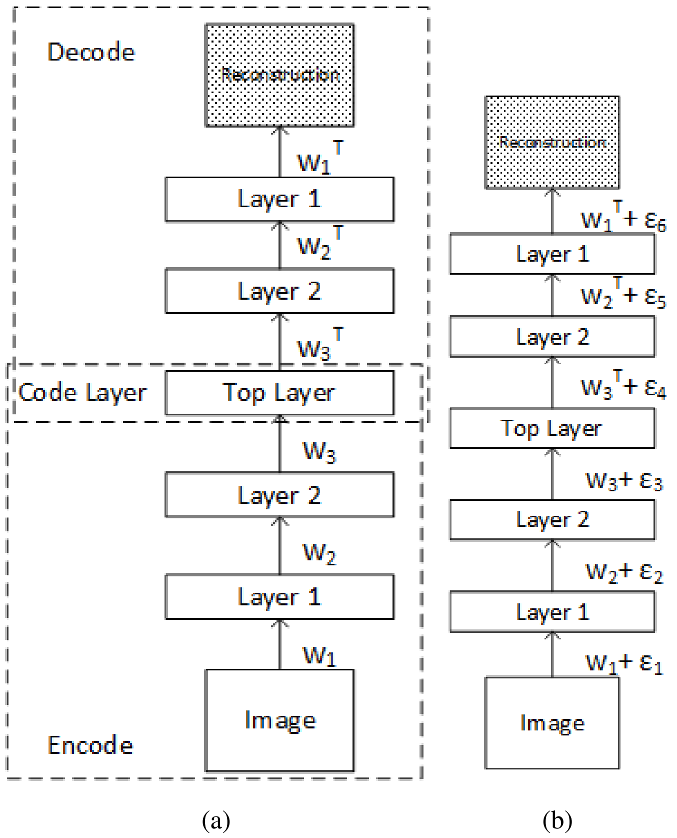


Fig. 2. The unrolling of successive RBMs to form an autoencoder is depicted in (a) with the middle layer being the code layer and the layer of interest. The weights are fine-tuned (b) using traditional feed forward networks with back-propagation for weight updates.

D. *Extraction of Sources*

When we look at the code layer, we want to be able to identify different groups of features, or activated units, as belonging to an instrument. There is no way to predict which features will be activated for an instrument. It could also be the case that instruments share sub components and thus activate the same features. Fig. 3 depicts a theoretical code layer for just one instrument present in the music signal. If the signal consisted of a different single instrument, we would expect a completely different set of features to be activated.

To identify an instrument in the code layer, a sample of the instrument could be presented to the autoencoder and the results noted. For example, separation of multiple sources can the be done by taking an identified region as belonging to a single source (or instrument) and setting the rest of the code layer to zero. Further identification of sources can be done through classification techniques such as Support Vector Machines or clustering in the code layer. Clustering typically assigns an instance, or in our case a unit in the code layer, to a cluster. A specialization of clustering is soft clustering, where an instance is assigned to a cluster with a probability. Intuitively soft clustering could provide for the best identification and grouping of sources in the code layer.

The modified code layer, after clustering or extracting only the region of interest is propagated back through the network to reconstruct inputs belonging to the identified source. We hence have a TF representation only belonging to a single instrument and by applying inverse STFT we can reconstruct a signal.

Standardized performance measurements for source separation is discussed in [30], and could be applied to evaluate the quality of the source reconstruction. It effectively measures the interference, noise and artifacts in the reconstruction, allowing us to compare the results with other separation algorithms.

This completes the process of source separation through the use of an autoencoder. The TF representation contains powerful information but reducing the representation to only real values we lose some value information for reconstruction. Methods exist to reconstruct the signal regardless and we continue, using the spectrogram as input to an autoencoder. The autoencoder creates an abstract representation of the input and identifies features of the instruments in a completely

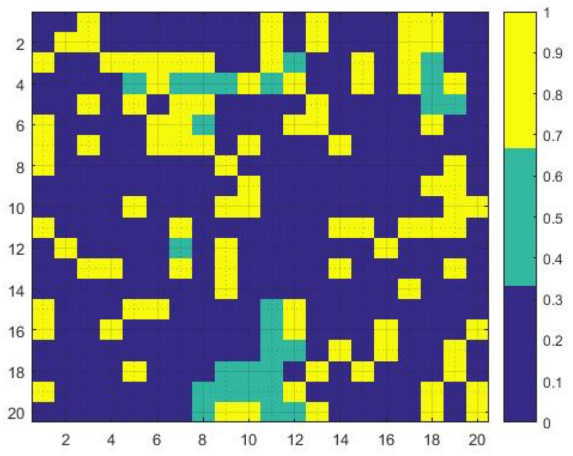


Fig. 3. A illustration of what the code layer could look like for a signal with only one source. The pixels correspond to activated features or partially activated features. For a different source instrument we expect different regions to be activated.

unsupervised training effort. Finally reconstructing a source involves identifying which features are activated in the code layer for a specific instrument. This unfortunately happens in a supervised way, but should be very flexible. Extracting instrument specific features allow us to reconstruct the inputs and subsequently reconstruct a time signal for a single source.

IV. Results

In this section we discuss the setup of experiments and the parameters used. Two tests were conducted to determine the validity of network as a source separation algorithm. Firstly, the reconstruction of the input produced by the network was visually compared to the input spectrogram. Secondly, the code layer separation was visually compared for different instruments to see if different features are activated in the code layer. No setup of the model has resulted in a clear separation of instruments and some clear deficiencies were identified. Only the single best set of results are discussed. More focus will subsequently be on possible improvements to the model in the next section.

A. Sample Creation

Audio samples are created using a set of five instruments. For each instrument individual notes or tones are taken and chronologically combined to form a sample of one instrument. These individual instrument samples are then uniformly linearly added to form a music sample. A subset of between 1 and 3 instruments are used to form a music sample. The five instruments are drums, flute, guitar, violin and piano samples taken from the Philharmonia Orchestra website [31]. Sample rate is the default 44100Hz, i.e. one second of music sample contains a vector of 44100 values. All music is converted to mono channel sound, so only one input vector is produced.

B. Parameter Selection

Here we will work through the parameter settings of the subsections as described in section III. A combination of parameters is difficult to determine as some parameters are more sensitive than others.

A music sample of twenty seconds was used as input to the STFT. Window length m is determined according to the bandwidth parameter, $m = 1.81(f_s/b)$, where f_s is the sampling frequency 44100Hz and b is the bandwidth. For visual evaluation of the spectrogram a value of $b = 150$ performs well, but a finer frequency resolution is achieved with $b = 75$ and is used to produce the spectrogram for input to the autoencoder. Further parameters settings for the STFT is less sensitive and the standard setup is adopted from [32]. A typical spectrogram can be seen in fig. 4. The 4 instruments sources are violin, piano, drums and guitar. With little experience it is easy to identify instruments in the spectrogram. The violin tones have vertically stacked squiggly lines. Piano tones are well defined for a frequency and start louder, fading away towards the end. Drums have sharp vertical lines which fade away quickly. Each pixel is one input unit to the autoencoder.

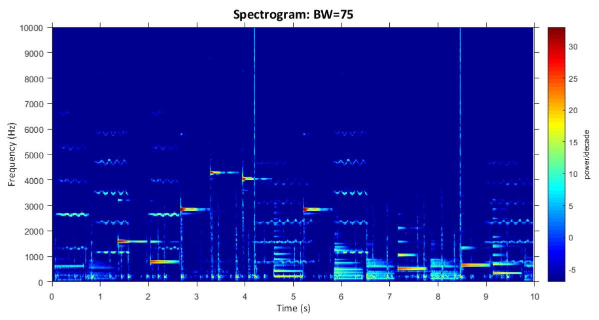
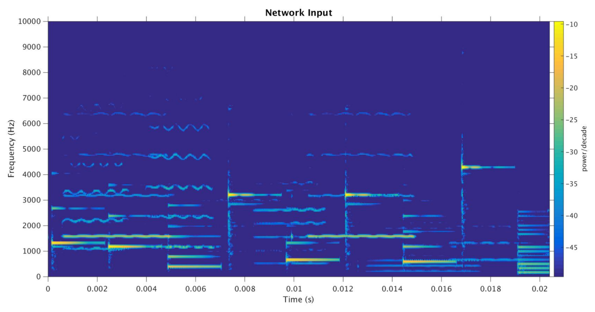


Fig. 4. A spectrogram of 4 instrument sources. $b = 150$

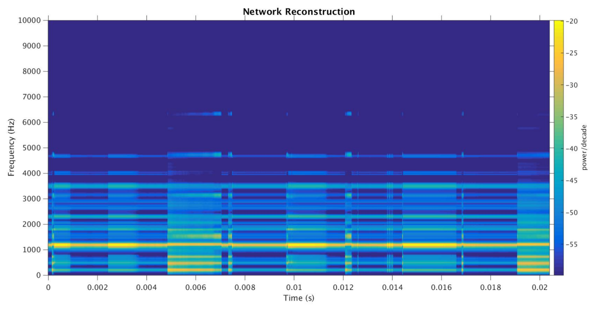
Once the input spectrogram is created, the next stage is to use it as input to the autoencoder. Chronological samples of 15 time frames are taken from the spectrogram as input to the autoencoder. The frequency resolution amounted to 533 samples. Gaussian binning was considered on the frequency dimension, but results showed that this is less effective and windowing is already performed in the STFT algorithm. An autoencoder with four stacked layers is used. The reason being the depiction of 4 levels, namely, factors, components, sources, and a mixture as in [1]. The weight vector is a convolution along the frequency access. To keep the layers sizes constant each layer has 500 hidden units (533 units in input layer). The activation function is the traditionally used sigmoid function. Although a rectified linear unit activation function has show better results in autoencoder training recently [18], the function was found to be incompatible with our network.

C. Input Reconstruction

Now we can evaluate the reconstruction of the input spectrogram produced by the autoencoder. Samples are stitched together to form a spectrogram which can be visually evaluated. Fig. 5 shows the result. The reconstruction (b) is not successful to the extend that individual features are unrecognizable, as opposed to the clear input in (a). Time distortion is apparent and the autoencoder fails to be sensitive to local time changes.



(a)



(b)

Fig. 5. (a) The input spectrogram and (b) the reconstructed spectrogram by the autoencoder. As we can see the reconstruction is not very representative of the input.

D. Feature Activation in the Code Layer

Next step is to evaluate if the code layer features are activated differently for different instruments. This is done by feeding 1000 samples of an instrument to the autoencoder, propagating the input spectrogram through to the code layer and averaging over the 1000 samples in the code layer. In fig. 6 we can see that there is no clear separation of sources in the code layer. This indicates a failure of the autoencoder to distinguish between different instruments. The inability to register and react to time fluctuations is also apparent, indicating that a time component is required for the autoencoder to function more effectively.

It is interesting to see in the result, the autoencoder does indeed register a slight difference for string instruments and air instruments, due to their radically different depictions in the spectrogram. This indicates that there might indeed be a room for improvement.

Reconstruction of separated sources proves to be trivial, since no adequate separation of sources could be observed. Effort was thus focused on finding the best possible combination of network structure for the current setup as shown in the results. Further study was performed on possible improvements to the autoencoder as a separation technique, or rather why the autoencoder is not the best suited to this particular problem. A discussion thereof follows in the next section.

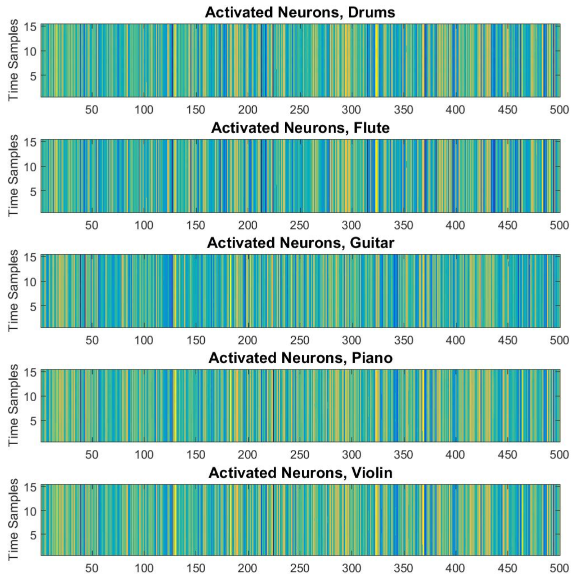


Fig. 6. The activation of the code layer for each instrument. The coloring of activated units corresponds to that of fig. 3. The instruments in order are drums, flute, guitar, piano and violin. Overall there is not much difference for activation, but slight changes are apparent. We can see piano and violin are more similar, while flute and drums are more similar. This is due to how they are represented in the spectrogram. A flute or drum is quite noisy, while piano and violins have a more apparent harmonic structure.

V. Further work and improvements

Here we investigate possible improvements and alterations that can be implemented to improve the performance of the autoencoder. The parameter space is discussed with a brief mention how NMF can be used as initialization to the autoencoder. A look at alternative TF representations such as the wavelet transform. A further brief study of the development of complex-valued autoencoders and possible implementations. Alternative ANN structures are discussed, other network structures have been used with success in speech processing problems.

A. Investigating the parameter space

From simulations it is apparent the structure of the autoencoder has little effect on the results. This could be entirely due to the autoencoder not being able to train on the spectrogram input, or rather, the properties of the spectrogram pixels as inputs. What is apparent from fig. 6 is the unnecessary information. The feature activation is too similar for all instruments, indicating that compression, i.e. reducing layer sizes, could lead to better results. This was of course experimented with, but without meaningful input parameter to the autoencoder, reducing layer size had little effect.

The inability of the autoencoder to learn small time patters indicates that the number of samples given to the autoencoder needs to be experimented with. Quite possibly further reducing the bandwidth b, resulting in a higher frequency resolution and lower time resolution, could improve the capability of the autoencoder. Smaller time scales should also be investigated, even unto the point of individual time samples vectors taken from the spectrogram. The reasoning here being that the STFT algorithm already performs windowing over the time axis, as well as over the frequency axis.

Alternatively, the axis along which the autoencoder learns, the convolution axis, could be exchanged. Currently the autoencoder performs along the frequency axis, keeping the time resolution constant. Experimenting with transpositions of the spectrogram could enable the network to overcome the initial hindrance of time dependence.

Finally, an initialization layer could be added in the form of NMF representation V_i, see (1). The array V_i is then an estimated of source i and used as inputs to the autoencoder in the form of a spectrogram. This is of course already done in [14], but it will provide a good baseline for what our autoencoder is capable of.

B. Alternative time-frequency representations

Wavelet transforms have been used with increasing success. It has a key advantage over Fourier transforms in that it captures both frequency and time information. A Fourier transform requires windowing to represent time, as is observed in the discussion of the STFT in section III-A. Wavelet transforms also excel at compression of signal data, although this not necessarily a desired property in our case. We want to preserve as much information about the signal as possible. A wavelet transform has an image representation called a scaleogram or scalogram. The scaleogram is analogous to the spectrogram for Fourier transforms.

A distinct advantage is that wavelet transforms, in most cases, do not produce complex-valued outputs. This is perfect for the inputs of a autoencoder. There is however a drawback, algorithms for wavelet transforms that are invertible, still produce complex-valued outputs and are only defined in the continuous transform case. The Fourier transform is used as an intermediate in the invertible wavelet transform algorithm, and as such complex values are the result.

This does not exclude the use of wavelet transforms to produce inputs to the autoencoder, rather, the advantage of using a wavelet transform over a Fourier transform becomes trivial. The inclusion of time information directly in the transform could still prove to be valuable in the retention of information during the separation process.

C. Complex-Valued Autoencoders

The studies on complex-valued inputs for autoencoders is decidedly limited. An extensive discussion of the possible implementation of complex values as input to autoencoders is done by [33], although the algorithm development focuses specifically on linear autoencoders. Linear autoencoders imply the transformation between layers are linear, non-linear activation functions cannot be used. The autoencoder is less able to learn complex relations between layers, a specific requirement for source separation. Despite the linear transformations, the algorithm only differs in the error function:

$$\begin{aligned}\min E(w_i, w_j) &= \min_{w_i, w_j^T} \sum_{t=1}^{m} ||x_t - w_i w_j(x_t)||^2 \\ &= \sum_{t=1}^{m} (x_t - \bar{w_i} w_j(x_t))\,(x_t - w_i w_j(x_t))\end{aligned} \tag{7}$$

where x_t is the inputs to the autoencoder, w_i and w_j is the corresponding weight vectors, similar to our case where we have the weight vectors w_i and w_i^T as in fig. 1(a). In (7) is the more general form of an autoencoder where the weight vectors for encoding and decoding layers are not taken as the same. The complex conjugate is signified by $\bar{\cdot}$. It is suggested the reader observes [33] as a full discussion and proof of derivation is given. It is also important to note that training the autoencoder happens slightly different, since the function landscape is considerably larger.

Another incorporation of complex values is done through tied input weights. This is introduced in [25], [34]. The process performs by combining cross products and inner products to create an input matrix. It has been implemented with success to identify similar pairs of images which has undergone slight rotations. In our approach this of course raises the question how will the original complex-valued inputs be reconstructed. Although not impossible, this remains to be experimented with.

D. Alternative ANNs to Autoencoders

The inability of an autoencoder to effectively translate time variation to source separation is a result of how the network inherently functions. The network should be able to incorporate small shifts in time and frequency and be sensitive to these shifts. It could alternatively incorporate a memory feature which allows the network to learn time varying representations. What is described here is firstly, a Convolutional Neural Network (CNN) and, secondly a Recurrent Neural Network (RNN). Both have recently gained popularity in speech processing problems with the advances made in CNNs by [13] and in RNNs by [4], [24].

CNNs process the inputs in small localized parts, looking for relevant features. Successive blocks are processed by making small shifts in time and/or frequency axis. The outputs of CNNs can be used as the input to other networks, such as autoencoder, again allowing for layers to be stacked and a "deep architecture" to be achieved. This approach could help the autoencoder learn time shifts.

RNNs operate by using outputs from the network in the previous time-step as inputs to the next time-step. This can be further expanded by delaying the number iterations before the current output is used again, creating a memory effect. It introduces the Long Short-Term Memory architecture. In [24] such a RNN was shown to outperform all algorithms.

The success of both these networks as an application in speech processing indicates that ANNs definitely have the capability to separate sources, although the setup of the inputs are very important and network structure should be able to incorporate shifts in time.

VI. Conclusions

Audio source separation is a very interesting problem with many collaborative facets and a variety of algorithms, each with advantages and disadvantages. Although speech processing has received considerable more attention and development, music source separation methods can borrow ideas and algorithms from the advances in speech processing. The complete workings of the process to separate audio sources was discussed and simulations with audio samples provided an overview of the ineffectiveness of the naive autoencoder approach. The naive approach to audio source separation did not prove to be successful and a more dedicated audio separation process should be implemented. The autoencoder was unable to effectively deal with the time variation in input spectrograms. The reasons and possible improvements are discussed, a convolutional approach to time incorporation is definitely worth considering. Reconstruction of the sources after separation was discussed, although not implemented, since the initial results proved that the autoencoder could not find any meaningful separation of sources. Reconstruction will be much easier if complex values are used as input. Methods to do this is discussed as part of possible improvements. Overall, this research proved source separation to indeed be a very difficult and multifaceted problem. A perfect separation into constituents will always be elusive due to the lack of complete information.

Acknowledgment

My journey the last two years as part of the Erasmus Mundus Complex System Science Master's Program has proved to be a success in more than just a knowledge experience. The European Consortium deserves special thanks for the funding and opportunity provided by the program. Finally the respective staff, lecturers and supervisors who had to put in extra hours to accommodate the Erasmus Mundus students deserve all the praise in the world.

References

[1] A. Ozerov, E. Vincent, and F. Bimbot, "A general flexible framework for the handling of prior information in audio source separation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 4, pp. 1118 – 1133, May 2012, 16. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00626962

[2] E. Grais and H. Erdogan, "Single channel speech-music separation using matching pursuit and spectral masks," in *Signal Processing and Communications Applications (SIU), 2011 IEEE 19th Conference on*, April 2011, pp. 323–326.

[3] F. R. Bach and M. I. Jordan, "Blind one-microphone speech separation: A spectral learning approach," in *Advances in Neural Information Processing Systems 17*, L. Saul, Y. Weiss, and L. Bottou, Eds. MIT Press, 2005, pp. 65–72. [Online]. Available: http://papers.nips.cc/paper/2572-blind-one-microphone-speech-separation-a-spectral-learning-approach.pdf

[4] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8599–8603.

[5] N. Jaitly and G. Hinton, "Learning a better representation of speech soundwaves using restricted boltzmann machines," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5884–5887.

[6] N. Murata, S. Ikeda, and A. Ziehe, "An approach to blind source separation based on temporal structure of speech signals," *Neurocomputing*, vol. 41, no. 1âĂŞ4, pp. 1 – 24, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231200003453

[7] Y. Wang and D. Wang, "Cocktail party processing via structured prediction," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 224–232. [Online]. Available: http://books.nips.cc/papers/files/nips25/NIPS2012_0128.pdf

[8] S. T. Roweis, "One microphone source separation," in *In Advances in Neural Information Processing Systems 13*. MIT Press, 2000, pp. 793–799.

[9] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 3, pp. 1066–1074, March 2007.

[10] T. Kohonen, "An introduction to neural computing," *Neural networks*, vol. 1, no. 1, pp. 3–16, 1988.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[12] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2009, pp. 545–552.

[13] L. Tóth, "Combining time-and frequency-domain convolution in convolutional neural network-based phone recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 190–194.

[14] E. M. Grais, M. U. Sen, and H. Erdogan, "Deep neural networks for single channel source separation," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3734–3738.

[15] D. E. Rumelhart, J. L. McClelland, P. R. Group *et al.*, "Parallel distributed processing, vols 1 and 2," *Cambridge, MA: The MIT Press*, 1986.

[16] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google

[17] S. Arberet, A. Ozerov, R. Gribonval, and F. Bimbot, "Blind spectral-gmm estimation for underdetermined instantaneous audio source separation," in *Independent Component Analysis and Signal Separation*, ser. Lecture Notes in Computer Science, T. Adali, C. Jutten, J. Romano, and A. Barros, Eds. Springer Berlin Heidelberg, 2009, vol. 5441, pp. 751–758. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00599-2_94

[18] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8609–8613.

[19] J. Cardose, "Blind source separation: Statistical principles," in *IEEE Proc*, vol. 9035, 1998, pp. 2009–2026.

[20] T. Virtanen, "Sound source separation using sparse coding with temporal continuity objective," *Proc. ICMC*, vol. 3, pp. 231–234, 2003.

[21] F. C, J.-L. Bertin N FAU Durrieu, and D. JL, "Nonnegative matrix factorization with the itakura-saito divergence: with application to music analysis." CNRS-TELECOM ParisTech, 75014 Paris, France. fevotte@telecom-paristech.fr FAU - Bertin, Nancy, pp. –.

[22] D. P. Ellis, "Prediction-driven computational auditory scene analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 1996.

[23] "Url." [Online]. Available: http://bass-db.gforge.inria.fr/fasst/

[24]

[25] A. Droniou and O. Sigaud, "Gated autoencoders with tied input weights," in *International Conference on Machine Learning*, 2013, p. x.

[26] J. R. Fienup, "Reconstruction of an object from the modulus of its fourier transform," *Optics letters*, vol. 3, no. 1, pp. 27–29, 1978.

[27] ——, "Reconstruction of a complex-valued object from the modulus of its fourier transform using a support constraint," *JOSA A*, vol. 4, no. 1, pp. 118–123, 1987.

[28] P. L. Van Hove, J. S. Lim, and A. V. Oppenheim, "Signal reconstruction from fourier transform amplitude," in *26th Annual Technical Symposium*. International Society for Optics and Photonics, 1983, pp. 214–225.

[29] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.

[30] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 1462–1469, 2006.

[31] "Url." [Online]. Available: http://www.philharmonia.co.uk/explore/make_music

[32] "Url." [Online]. Available: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

[33] P. Baldi, S. Forouzan, and Z. Lu, "Complex-valued autoencoders," *CoRR*, vol. abs/1108.4135, 2011. [Online]. Available: http://arxiv.org/abs/1108.4135

[34] R. Memisevic, "On multi-view feature learning," *CoRR*, vol. abs/1206.4609, 2012. [Online]. Available: http://arxiv.org/abs/1206.4609