Signal reconstruction by means of Embedding, Clustering and AutoEncoder Ensembles

Corrado Mio[2], Gabriele Gianini[1,2]
[1] *EBTIC/Khalifa University of Science and Technology, Hadbat Al Zaafran, 127788 Abu Dhabi, UAE*
[2] *Dipartimento di Informatica, Università degli Studi di Milano, via Celoria 18, 20144 Milano (MI) - Italy*
Email:{corrado.mio,gabriele.gianini}@unimi.it

Abstract

We study the denoising and reconstruction of corrupted signals by means of AutoEncoder ensembles. In order to guarantee experts' diversity in the ensemble, we apply, prior to learning, a dimensional reduction pass (to map the examples into a suitable Euclidean space) and a partitional clustering pass: each cluster is then used to train a distinct AutoEncoder. We study the approach with an audio file benchmark: the original signals are artificially corrupted by Doppler effect and reverb. The results support the comparative effectiveness of the approach, w.r.t. the approach based on a single AutoEncoder. The processing pipeline using Local Linear Embedding, k-means, then k Convolutional Denoising AutoEncoders reduces the reconstruction error by 35% w.r.t. the baseline approach.

1. Introduction

During propagation from the source to the receiver, a signal is affected by several physical phenomena that degrade its quality, such as multiple reflection, refraction, diffraction and absorption; if the source is moving relatively to the receiver, the signal will be affected also by the Doppler effect. An expert wishing to design an algorithm or a system intended to correct the received signal for the effect of the channel has to know the parameters of the different mechanisms, or to have a way for probing the properties of the channel. Even so, the task of designing a flexible enough algorithm for the signal restoration (generically called ”denoising”) is extremely demanding, in terms of modeling complexity.

However, if a sample of transmitted and received signal pairs is available, one can resort to Machine Learning techniques: one can learn from the data a model of the inverse of transfer function of the channel, then use it to reconstruct an approximation of the transmitted signal.

The approach has already been used in a work [1] applied to the radio spectrum. The approach, however, is very general and can in principle be used with any EM communication spectra ad also with acoustic communications. One of its key elements is that the time-domain signal is mapped into a spectrogram, using the Short Time Fourier Transform, then the spectrogram is treated as an image.

The approach used in [1] consists in applying a standard Deep Learning technique: training a Convolutional Denoising AutoEncoder (CDAE) to denoise the signal (the convolutional layers are adopted to exploit at best the locality of the ”image”). The CDAE, in that work, encodes the inverse of the transfer function and allows to effectively remove the effect of the channel for a limited number of prototype input signals (there, two different types of WiFi protocol preambles). This is equivalent to denoising a small variety of symbols (in [1], two symbols), in presence of variable ”noise” conditions.

The point is that by increasing the variety of input symbols, a single CDAE looses effectiveness. This hints at a natural approach allowing to scale up the denoising to a higher complexity: using an ensemble of CDAEs. Each CDAE can be trained to become ”expert” of a region of the configuration space for the incoming signal: in principle, by assigning to the responsible expert the task of denoising the incoming signal the reconstruction quality can be improved.

In order to train experts and reduce their co-evolution, a number of techniques are available. We study of the successive application of 1) a dimensional reduction and embedding step (using techniques from [2] or [3]) and 2) a k-means partitional clustering step [4], to identify a number of sub-samples 3) to train a number k of CDAE, each on a different cluster. We compare this technique to the use of a single CDAE (without embedding and clustering) and show that our approach partially improves the quality of the reconstruction.

The paper is structured as follows: in Section 2 we describe the methodology, in Section 3 the data set preparation and analysis, in Section 4 we discuss the results.

2. The methodology

We work on spectrograms, treating them as if they were images, with the objective of removing the effects that altered the original signal. We refer, conventionally, to this task as denoising. From one original audio signal we obtained altered signals by Doppler effect, by reverb in a small-hall and in a large-hall scenario and by the combination of the reverb scenarios with the Doppler effect.

A single Denoising AutoEncoder could be used for the task, but with a large number of different signals to reconstruct, the capacity of an AutoEncoder can be challenged. For this reason we set up a procedure to share the task among an ensemble of AutoEncoders.

2.1. Ensemble Learning and expert diversity

There is a rich literature on ensemble learning. This is a process by which multiple models, such as experts/algorithms, are suitably generated and combined to solve a problem: the combination of the experts in the ”committee” yields better performance and stability guarantees w.r.t. the ones provided by a single expert/ algorithm in isolation. A well-known representative of the ensemble learning algorithms is the Random Forest algorithm [5], [6], others are the Bagging algorithm [7], Boosting algorithms, such as AdaBoost [8] and the stacking methods [9].

A common issue, in the optimization of an ensemble based algorithm, is how to create diversity during the learning phase, among the experts of the committee. For instance, Bagging algorithms train each expert on a different bootstrap sample of the training data, before aggregating them, e.g. by majority vote. Random Forest increases further the diversity by training each tree of the forest on a different set of attributes, chosen at random. In the present work, we use a committee of denoising AutoEncoders, and we adopt the following strategy for promoting diversity: we have the AutoEncoders trained on different subsets of the training dataset; the subsamples are chosen by clustering the points (each representing a sample), after a dimensional reduction and embedding step.

2.2. The phases of the procedure

The phases of the procedure are, therefore, three: low dimensional embedding, partitional clustering and AutoEncoder training.

2.2.1. Low-dimension Embedding. The purpose of the low dimensional embedding is to project the data into a low dimensional Euclidean space, on which, later, a clustering algorithm based on a metric distance can but put at work. For such an embedding one can, in principle, try several forms of Nonlinear Dimensionality Reduction algorithms: we experimented with t-SNE (Student t Stochastic Neighbor Embedding) [3] and LLE (Locally Linear Embedding) [2].

The LLE algorithm is used to identify low-dimensional local structures in high dimensional data, based on the neighborhood of the points: it characterizes the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors, preserving the local structure (what is close in the starting space, remains close in the low-dimensional space). It works by minimizing an embedding cost function defined using locally linear reconstruction errors.

While the embedding LLE can be justified essentially based on geometric concepts and metric distances, the embedding by t-SNE is based on probabilistic approach and uses similarities. The algorithm t-SNE is a variant of SNE (Stochastic Neighbor Embedding) [10]. The latter starts by transforming high-dimensional distances between data points into conditional probabilities representing similarities. The similarity of a data point to another is defined as the conditional probability that the former would pick the latter from its neighborhood, if neighbors were picked in proportion to their probability density under a Gaussian centered at the former point. The algorithm t-SNE replaces the Gaussian by a t-Student density, improving the performance and the desired separation between dissimilar points.

Both the algorithms LLE and t-SNE, take a high dimensional sample of points and map it onto a lower dimensional manifold. We chose to experiment with dimensions $d = 2$ and $d = 3$.

2.2.2. Partitional clustering. The purpose of the partitional clustering algorithm is to break the whole sample in regions of interest, on each of which a distinct expert denoiser can be trained. Several partitional clustering algorithms are obviously up to the task: since we are not exceedingly interested in achieving a high quality for the clustering – provided that it is able to yield a reasonable division of labor for the AutoEncoders – we choose to experiment straightforward algorithms of the k-means family: k-means and k-medoids.

The problem solved by the family of methods of the k-means algorithm is the following. Given m instance descriptions $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$ and chosen a number k of clusters in which the set of instances has to be partitioned, find a partition that minimizes a predefined intra-cluster dissimilarity. Instances in the same cluster will be more similar to each other than instances in different clusters (according to the pre-defined criterium of similarity). The algorithms of this family formalize the intra-cluster similarity in terms of the dissimilarity to a centroid point of the cluster. The algorithms of the family start from a tentative partitioning and at each iteration recompute the centroid and reassign each object to the cluster with the closest centroid. The definitions of the centroid and of the dissimilarity between an object of the dataset and a centroid, determine the differences among members of this algorithm family. A special case occurs when all the elements of a row description are numerical: in this case one can naturally choose as a dissimilarity measure a metric distance. This is the choice adopted in k-means, which defines the cluster centroid as the center of mass of the cluster. Other choices for the centroid are the mode of the cluster (k-modes) or the element of the cluster close to the center of mass (k-medoids).

The algorithms of the k-means family get in input a list of points and yield in output a partition of those points in k clusters, where the integer k is a parameter of the algorithm. The value of k we chose was note high: our purpose was to pass from the benchmark setting – with only one AutoEncoder – to a setting endowed with higher capacity, so we experimented with the lowest k's able to provide extra accuracy. The results shown in the following section focus on the choice $k = 3$.

2.2.3. Denoising AutoEncoders. AutoEncoders (AEs) [11], are multilayered Artificial Neural Network algorithms characterized by a special training procedure and a symmetric input-encoding-output architecture. They learn by self-supervision, i.e. try to replicate the input example on the output: each input record is its own structured label. The minimal standard architecture of an AE consists of an input layer, a hidden layer, and an output layer. AEs are trained using the back-propagation algorithm. After the training their learned weights are frozen, and the layer(s) beyond the central (encoding) one are discarded. What is left is a function that maps the input into a different representation, whose expected properties depend on the details of the architecture and of the training. Denoising AEs [11] are trained by providing them a noisy input and challenging them to output, not the input itself, but a non-noisy version of it: i.e. they are forced to find a representation where the signal is unfolded from noise. In other words, De-noising AutoEncoders (DAEs) follow the general concept of AutoEncoder but use a noisy input and the reconstructed signal is compared to the clear signal in the loss function.

In our experimentation, we used AEs with convolutional layers (i.e. we used a Convolutional De-noising AutoEncoder, or CDEA): in those architectures, parameters of a patch of the image are shared among all the locations to enforce spatial locality. Convolutional AEs perform better than classical AEs in image processing [12].

3. Datasets and data preparation

We illustrate the methodology using as a starting point a demonstrative data set of snippets, chosen for the variety of spectra: the **wabrlrb** dataset [13].

3.1. The Dataset

The **wabrlrb** dataset consists of $10,000$ audio files, sampled at $44,100$ Hz, each of approximately 10 seconds duration (this was also the maximum duration, when the duration was shorter we padded the interval with silence: eventually all the samples had the same duration). The spectrogram of an audio sample is shown in Figure 1.

We altered the original audio by introducing some *reverb* and some *Doppler effect*. The tool used for the data editing and for the subsequent analysis is the Wolfram's Mathematica environment. In Mathematica, the reverb effect (*AudioReverb*) is based on different environment models: we experimented with the *SmallHall* and *LargeHall* environments. The Doppler effect has been simulated by applying a multiplicative factor to the frequencies (the pitch), shifting them upward or downward. In a recurrent Doppler effect scenario, the source approaches the receiver

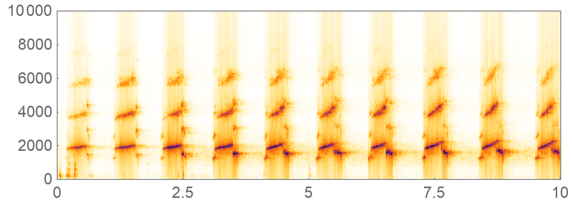


Figure 1. Spectrogram (frequency (Hz) vs. time (secs)) of one of the audio samples: the darker the color the higher the modulus of the corresponding complex coefficient. The 2D matrix of the spectrogram consists in 1294×512 elements, i.e. pixels.

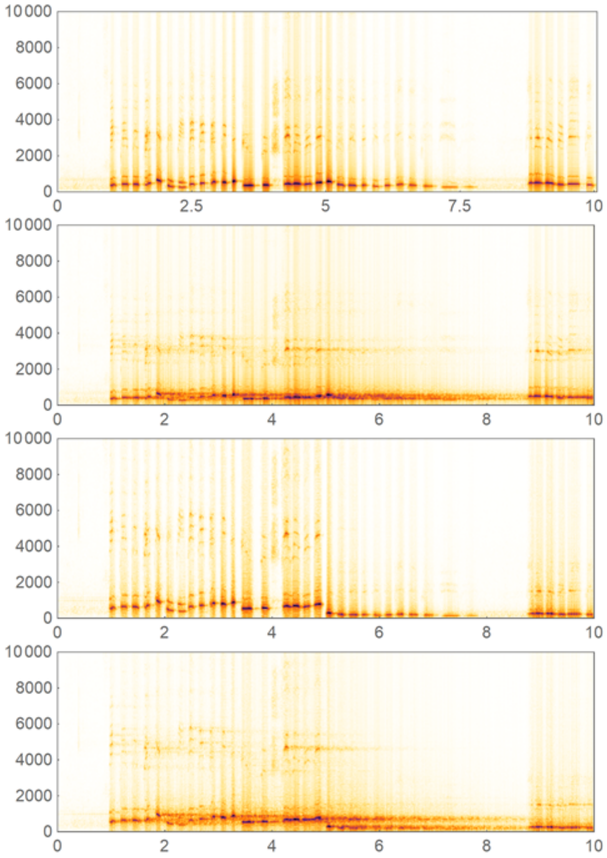


Figure 2. A spectrogram and its altered versions. From top to bottom: the original signal; the signal with reverb only in a LargeHall configuration; the signal with Doppler effect only; the signal with Doppler effect and LargeHall reverb.

at constant velocity, reaches a minimum distance, then it starts getting farther from the receiver. Given the source own frequency (original pitch), this determines first a higher frequency (higher pitch) then a lower frequency (lower pitch): we emulated this scenario, breaking the original signal in two and choosing a pair of multiplicative factors, one higher, and one lower than one, by which to multiply the frequencies: we chose 1.5 and 0.50. Examples of spectrogram alterations are shown in Figure 2.

3.2. Spectrogram generation

The Short Time Fourier Transform (STFT) is a transformation obtained applying the Fourier Transform (FT), not to the entire signal, but to windows from a certain sample: the results of each FT are juxtaposed so as to create

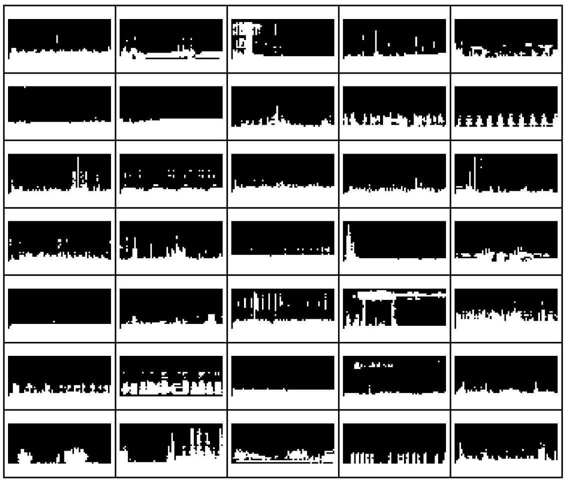


Figure 3. A sample of 35 binarized images of size 323×64.

a matrix. Each spectrogram represented the Short Time Fourier Transform (STFT) [14] of the raw time series $x(t)$ corresponding to the signal of a preamble. More specifically, a spectrogram pictures the STFT $S_x(\tau, \omega)$ as a function of the discretized time τ and discretized frequency ω. Each signal was partitioned into 1294 time intervals, for each interval 512 frequencies were computed. Thus, each image consisted in 1294×512 gray-level pixels, where the gray-level expressed the modulus $|S_x(\tau, \omega)|$.

The Fourier Transform is typically implemented using the Fast Fourier Transform (FFT). By construction, the FT is invertible, however the computation of the inverse of the STFT requires care, since the quality of the transform depends on the number of samples and is affected by the time quantization. In typical implementations, instead of shifting the moving window by an amount of time equal to its own size, one moves the window so as to have a certain time-overlap. Then, when the STFT is inverted, instead of using the entire reconstructed signal (relative to that window), only the non-overlapping part is used.

Due to this, there are two fundamental parameters that must be taken into consideration: the size of the window and the relative overlap. Since the FT is implemented using a discrete FFT, it is desirable that the size of the window is a power of 2. One has furthermore to reconcile contrasting requirements: with a narrower window, a more accurate temporal reproduction is obtained at the expense of the reconstruction quality in terms of frequencies; with a wider window, a more accurate reproduction in terms of frequencies is obtained at the expense of temporal reconstruction; a small overlap increases calculation times without necessarily increasing the quality of the reconstruction. Overall, we chose each spectrogram matrix to have $size = height \times width$ of 1294×512 elements, i.e. pixels. The implementation made available by Mathematica makes the following choices. The window size is $n = 2^{\lfloor 1+log_2\sqrt{m} \rfloor}$, where m is the number of samples; the window displacement is $d = \lfloor n/3 \rfloor$, that is the windows overlap by $2/3$ of their size.

Obviously, all the above operations can create small artifacts in the spectrogram, however, for the purposes of our alteration and reconstruction procedure they are not relevant. This holds also because in order to reduce the complexity of the computation, before applying embedding and clustering, the spectrograms are compressed by binarization. Indeed each uncompressed spectrogram considering 256 distinct gray levels would fit into an array of size 1.3 Mb: manipulating arrays of this size is impractical. For this reason, the intensity value at each element is replaced by 0 or by 1 depending on its position w.r.t. the mean of the image. More precisely, the values of $|S_x(\tau, \omega)|$ have been quantized and encoded with binary values $b_x(\tau, \omega)$ as follows: $b_x(\tau, \omega) = 1$ if $|S_x| > \langle S_x(\tau, \omega) \rangle$ (the average $\langle \cdot \rangle$ is taken over the spectrogram pixels) and zero otherwise. The overall spectrogram construction is similar to the one used in [1], except that we used the Mathematica implementation, and different time and frequency quantization parameters. We tried also to reduce the quantization in the number of frequencies from 512 to 128 and to 64 (re-scaling the other axis consequently), and did not see any appreciable change in the outcomes of the procedure, so we adopted the reduction to 64, saving considerable processing time.

A set of 35 binarized images of size 323×64 is shown in Figure 3 for illustration.

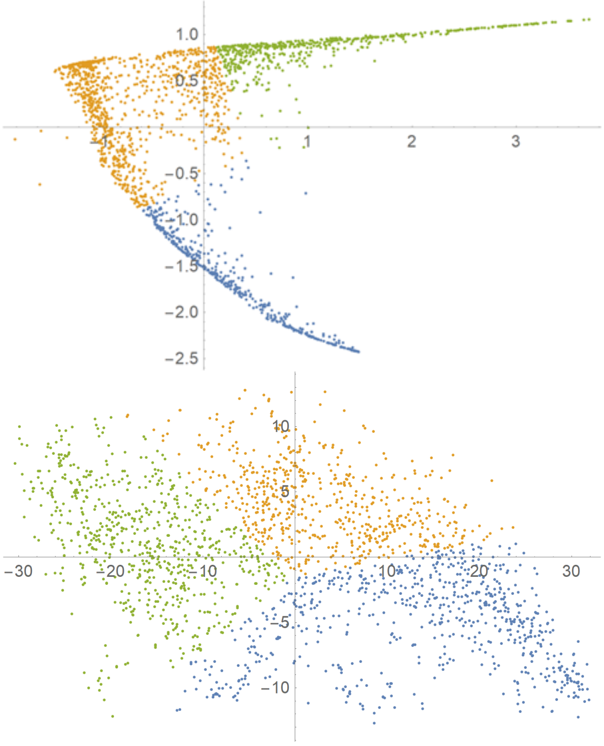


Figure 4. Yield of the k-means partitioning on the LLE embedding (top) and of the t-SNE embedding (bottom). In both cases $k = 3$.

4. Results and comparisons

4.1. Embedding and clustering

We tried a range of methods for dimensional reduction and embedding – available through the Mathematica libraries – focussing on LLE and on t-SNE. We tried both the dimensions $d = 2$ and $d = 3$. In the latter dimension the data points turned out to be organized approximately around a two dimensional manifold. Thus we decided to carry further our study using only $d = 2$.

We then combined those two embedding methods with the methods for partitional clustering, made available by Mathematica, focussing specifically on the algorithms of the k-means family, such as k-means, k-modes and k-medoids: we tried various values of k and of the stopping condition parameters.

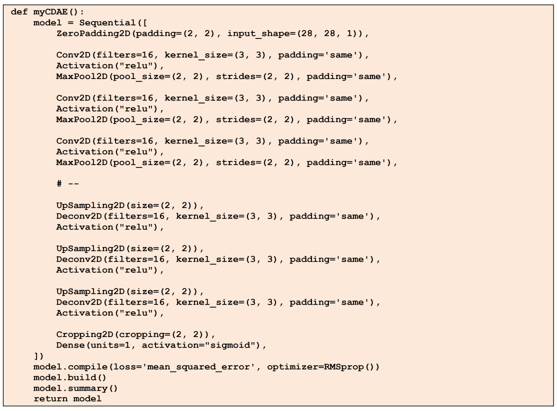
Figure 4 displays the results for both LLE and t-SNE after using k-means with $k = 3$. We delimited our subsequent experimentation with AutoEncoders to the case LLE, which presents a crisper cluster separation.

4.2. Denoising

We experimented with several variants of the architecture of the Convolutional AutoEncoders using different heterogenous datasets and aiming at performance and architecture simplicity. Eventually we chose the architecture defined in Figure 5.

We started from the $k = 3$ distinct datasets obtained by partitioning the whole dataset and trained a distinct CDAE on each dataset. Specifically: we gathered the IDs of the points in the same cluster in the two dimensional representation and put the corresponding spectrograms into the same training set. Then we trained the three CDAE. We used as a training set for each CDAE the 80% of the spectrograms of the respective cluster and left the remainder 20% for the test.

Ideally the procedure works as follows: a test spectrogram to be denoised is mapped by LLE or t-SNE into a two-dimensional point: the distance of this point from the cluster centers found by k-means determines the cluster



```
def myCDAE():
    model = Sequential([
        ZeroPadding2D(padding=(2, 2), input_shape=(28, 28, 1)),

        Conv2D(filters=16, kernel_size=(3, 3), padding='same'),
        Activation("relu"),
        MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='same'),

        Conv2D(filters=16, kernel_size=(3, 3), padding='same'),
        Activation("relu"),
        MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='same'),

        Conv2D(filters=16, kernel_size=(3, 3), padding='same'),
        Activation("relu"),
        MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='same'),

        # --

        UpSampling2D(size=(2, 2)),
        Deconv2D(filters=16, kernel_size=(3, 3), padding='same'),
        Activation("relu"),

        UpSampling2D(size=(2, 2)),
        Deconv2D(filters=16, kernel_size=(3, 3), padding='same'),
        Activation("relu"),

        UpSampling2D(size=(2, 2)),
        Deconv2D(filters=16, kernel_size=(3, 3), padding='same'),
        Activation("relu"),

        Cropping2D(cropping=(2, 2)),
        Dense(units=1, activation="sigmoid"),
    ])
    model.compile(loss='mean_squared_error', optimizer=RMSprop())
    model.build()
    model.summary()
    return model
```

Figure 5. The definition of the CDAE architecture used for the denoising.

to which the point has to be assigned, and, as a consequence, the CDAE by which it has to be denoised; the spectrogram is passed to the CDAE, and the output is compared to the transmitted version of the spectogram using a reconstruction fidelity metrics. As a metric ℓ we used the square deviation averaged over all the pixels of a spectrogram image.

Some representative example of reconstructed images is shown in Figure 6.

We used also as a baseline a CDAE with the same architecture, trained on the whole dataset (again 80% was used for trainig and the remainder the test).

The average $\langle \ell \rangle$ of this metric over the test sets is reported in Table I, both for the single-CDAE baseline-case and for each cluster of the three-CDAE case.

The Table is organized as follows: in the first data row one can find the information relative the baseline case involving the whole set of spectrograms (the Cl.Card. tells that whole data set contained $12,000$ elements, it is intended that out of those, 80% have been used for training, 20% for test, the same convention is used for all the other numbers of the same column). The reminder of the table is devoted to the three-CDAE approach. First are the rows describing the outcome of the use of LLE and k-means, then the lines relating to the use of t-SNE and k-means: the numbers in the column Cl.ID correspond to the ID of the cluster determined by k-means. The three rows referring to individual clusters are followed by a row containign the average of the performance metric over the three clusters.

One can see that the two different embedding techniques produce qualitatively similar results, and that globally our method outperforms the baseline. Specifically, the average performance of our approach (average over the $k = 3$ CDAEs) is never worse than the one of the baseline. Not all the three CDAE improve individually over the baseline, however, two out of three, i.e. the majority perform better (in blue the improving clusters, in red the others). This conclusion holds both with the square root of the mean square error and with the square root of the median square error. The use of the median here provides a more stable indicator with respect to outliers. Using the median-based error one can see an improvement of the 35% w.r.t. the baseline with LLE, and a 5% w.r.t. the baseline with t-SNE. Thus the embedding with LLE turns out to be neatly better.

5. Conclusions

We studied the denoising and reconstruction of a corrupted signal by means of an ensemble of AutoEncoders. In order to guarantee experts' diversity in the ensemble, we applied, prior to AutoEncoder training, a dimensional reduction pass (either with LLE or with t-SNE), that mapped the examples into a suitable low-dimensional Euclidean space; then we applied a partitional clustering pass. Each cluster has then been used to train a distinct AutoEncoder of the ensemble. For demonstrative purposes, we experimented the approach using a benchmark data set of audio files: we artificially corrupted the original signals introducing Doppler effect and reverberation. The results support the comparative effectiveness of the approach, w.r.t. an approach using a single AutoEncoder. Specifically, using LLE and k-means and three Convolutional Denoising Autoencoders, one achieves a 35% reduction of the reconstruction error w.r.t. the single CDAE case.

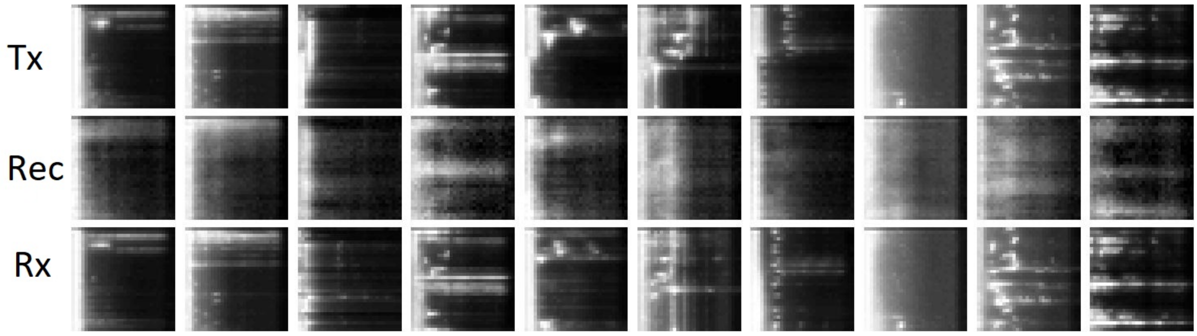


Figure 6. A sample of results. Top: Tx, transmitted signal. Bottom: Rx, received signal. Middle: Rec, reconstructed signal.

Table 1. Comparison between single CDAE and k=3 CDAEs reconstruction errors

Dimensional Reduction & Clustering	**Cl. ID**	**Cl. Card.**	**Root Mean Square Error** $\times 10^{-3}$	**Root Median Square Error** $\times 10^{-3}$
Baseline (single CDAE)	-	12000	**7.64**	**6.25**
LLE+ k-means (k=3) 3 CDAEs	1 2 3	6090 1204 4706	7.08 5.24 **9.79**	5.91 4.34 **8.48**
Average LLE			**6.65**	**4.05**
TSNE+ k-means (k=3) 3 CDAEs	1 2 3	3026 6660 2314	7.11 **9.14** 4.99	5.85 **7.91** 3.99
Average TSNE			**7.64**	**5.94**

Acknowledgements

The authors acknowledge the support of the ICT Fund at EBTIC/Khalifa University, Abu Dhabi, UAE. The work was partially funded by the EU H2020 Research Programme, within the projects Toreador (Grant Agreement No. 688797), Threat-Arrest (Grant Agreement No. 786890) and Concordia (Grant Agreement No. 830927).

References

[1] E. Almazrouei, G. Gianini, N. Almoosa, and E. Damiani, "A deep learning approach to radio signal denoising," in *(IWSS 2019) Smart Spectrum, 5th IEEE International Workshop*. IEEE, 2019.

[2] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[3] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[4] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[5] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[6] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[7] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[9] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[10] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Advances in neural information processing systems*, 2003, pp. 857–864.

[11] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[12] D. Lee, S. Choi, and H.-J. Kim, "Performance evaluation of image denoising developed using convolutional denoising autoencoders in chest radiography," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 884, pp. 97–104, 2018.

[13] "wabrlrb, http://dcase.community/challenge2018/task-bird-audio-detection."

[14] A. Mertins and D. A. Mertins, *Signal analysis: wavelets, filter banks, time-frequency transforms and applications*. John Wiley & Sons, Inc., 1999.