Mini-Project (ML for Time Series) - MVA 2024/2025

Martin Jolif martinjolif@gmail.com
Andrej Perković andrej.perkovic@etu.u-paris.fr

December 30, 2024

1 Introduction and contributions

We choose to study the following paper: "A review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling" [2] published in 2014. Developing robust features that capture relevant information can be beneficial to better model complex real-world data as time series data. However, developing domain-specific features for each task is expensive and requires expertise as well as labeled data which can be quite complicated to acquire. In this paper, the authors propose a review of several unsupervised feature learning algorithms for time series data. They very briefly present several methods like Restricted Boltzmann Machine (RBM), Auto-encoder, or Recurrent Neural Network (RNN). They cite a considerable number of other papers that used unsupervised features learning for solving task related to time series data like videos, music or physiological data.

They cite this paper: "Learning features from music audio with Deep Belief Networks" [1] which caught our attention. In this paper they present a Deep Belief Network (DBN) that learn features from music signal. Thanks to the features learned through the DBN, the authors solve a music genre classification task on the GTZAN dataset (jazz, rock, ...) as well as an autotagging task on the Majorminer dataset.

Based on this article, we decided to implement from scratch using PyTorch a Deep Belief Network to compare the results obtained by the authors from the article and results obtained from our own implementation to solve the genre classification task using the GTZAN dataset. To do so, we followed the processioning of the audio files described in the paper [1], and we use them to train our DBN. Secondly, at the end of the paper the authors suggest to explore hierarchical convolutional DBN. Going in this way, we explored two methods to try solving again the music genre classification task. In this way, we implement from scratch using PyTorch a convolutional RBM and a Variational Auto-Encoder with convolution layers. This time we didn't follows the data prepossessing presented in the paper and "created" another prepossessing adapted to convolutions.

For the organization and repartition of works, we firstly read the original [2] article and decided to focus on one specific topic and method to avoid getting scattered. At this point, we found interest in working on the paper presented shortly before. We first focused on the implementation of the DBN, therefore we created a GitHub repository to organize our work, before trying to explore the convolution part. We planned as far as to experiment with stacked convolutional networks for feature extraction, but we were discouraged by the subpar results of certain methods.

Martin Jolif contributed to the analysis of the DBN and convolutioanl VAE, while Andrej Perković contributed to the anbalysis of the convolutional RBM.

2 Method

2.1 Restricted Boltzman Machine (RBM) architecture

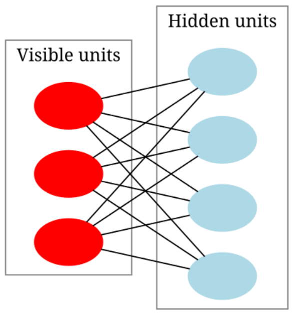


Figure 1: Restricted Boltzmann Machine with three visible units and four hidden units (no bias units)

The joint distribution for a given visible and hidden vector is the following:

$$\mathbb{P}(v,h) = \frac{1}{Z} e^{E(v,h)} \tag{1}$$

where $E(x,h) = h^T W x + b^T h + c^T x$ and Z a partition function that ensures that the distribution is normalized.

The individual activation probabilities are given by (we suppose the visible and hidden units to be binary):

$$\mathbb{P}(h_j = 1|x) = \sigma(b_j + \sum_i W_{ij} x_i) \text{ and } \mathbb{P}(x_i = 1|h) = \sigma(c_i + \sum_j W_{ij} h_j) \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid activation function. Then for the training process, the parameters W, b and and c are optimized by using the contrastive divergence algorithm:

Algorithm 1 Training

```
1: for all training sample $x \in X$ do
2:     $h \sim \mathbb{P}(\cdot|x)$
3:     $x' \sim \mathbb{P}(\cdot|h), h' \sim \mathbb{P}(\cdot|x')$
4:     W += $\alpha$ ($xh^T - x'h'^T$)                ▷ $\alpha$ is the learning rate
5:     b += $\alpha$ (h - h')
6:     c += $\alpha$ (x - x')
7: end for
```

where xh^T and $x'h'^T$ are called respectively positive gradient and negative gradient.

2.2 Convolutional RBM

The Convolutional Restricted Boltzmann Machine (ConvRBM), as outlined in [3], introduces several key concepts to extend the standard RBM for applications in image processing and object recognition. The idea in question was to use the Mel spectrogram of the time series as the "image" representation and then test out how would ConvRBM perform in the classification task. At the end, we would be interested in replicating the idea of stacking present in Deep Belief Networks in combination with ConvRBM.

Unlike standard RBMs, which connect all visible and hidden units, ConvRBMs use localized connections to exploit the spatial structure of images. These local connections mimic convolutional layers in neural networks. The hidden units in a ConvRBM are organized into feature maps. Weight sharing is implemented within each feature map, allowing the model to detect the same feature across the entire image. The conditional probabilities for hidden and visible units are derived based on local connections.

The energy function for a ConvRBM is defined as:

$$E(v,h;\theta) = -\sum_{k,q} h_{kq} \left(\mathbf{w}_k^T \mathbf{v}(q) \right) - \sum_i b_i v_i - \sum_{k,q} c_k h_{kq}$$

where v is the isible layer (input image pixels), h_{kq} is the binary hidden unit in the k-th feature map at position q, $\mathbf{w}_k$ are the filter weights shared across the k-th feature map, $\mathbf{v}(q)$ is the local receptive field of the visible layer at position q, b_i is the bias for visible units and c_k for hidden units in feature map k.

Contrastive Divergence (CD) is used for training, with modifications to account for the spatially constrained connections and to ensure consistency in handling boundary pixels. Features learned by ConvRBM are sparse and shift-invariant, which improves generalization and reduces redundancy.

Using Contrastive Divergence (CD), the parameter update rule becomes:

$$\Delta \mathbf{w}_k = \eta \left(\langle v(q) h_{kq} \rangle_{\text{data}} - \langle v(q) h_{kq} \rangle_{\text{model}} \right),$$

where η is the learning rate, $\langle \cdot \rangle_{\text{data}}$ is the expectation under the data distribution, $\langle \cdot \rangle_{\text{model}}$ the expectation under the model distribution.

In our implementation, we used a simplified version of CD. We opted for reconstruction-based training. It is simplified and does not directly compute the exact gradients for weights as theoretically required.

2.3 Deep Belief network

A deep belief network (DBN) can be seen as the composition of RBMs where each sub-network's hidden layer serves as the visible layer for the next RBM:

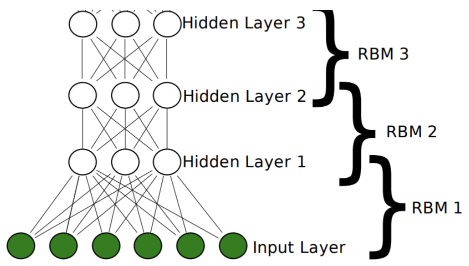


Figure 2: Deep belief network composed of 3 RBMs

For the training of a DBN, once the first RBM is trained (on the whole training set), another RBM is "stacked" atop it, taking its input from the final trained layer. The new RBM is then trained with the contrastive divergence algorithm as explained before.

Finetuning final layer To solve the music genre classification task, the authors of the paper [1] "stacked" a final classification linear layer atop this DBN and "fine-tune" it by initializing the RBMs weights with the pretrained weights and finally train the full network in a supervised manner. The pretrainig of the RBMs is relatively quick. However the final supervised training is very slow. We also tried to just pretrain the RBMs to get some features of the music samples and then use classical methods to do classification like SVMs or Logistic Regression. Our final idea was to aggregate several convolutional RBMs to create a convolutional DBN. However, since our simple convolutional RBM doesn't seems to provide good results we abandoned the idea.

2.4 Variational Autoencoder

A variational autoencoder is composed of two part: an encoder and a decoder. The encoder is a neural network that try to predict the mean and the covariance matrix of the data distribution in the latent space (a smaller dimensional space that the input space). The decoder try to reconstruct as much as possible the latent data to be the same the input data.

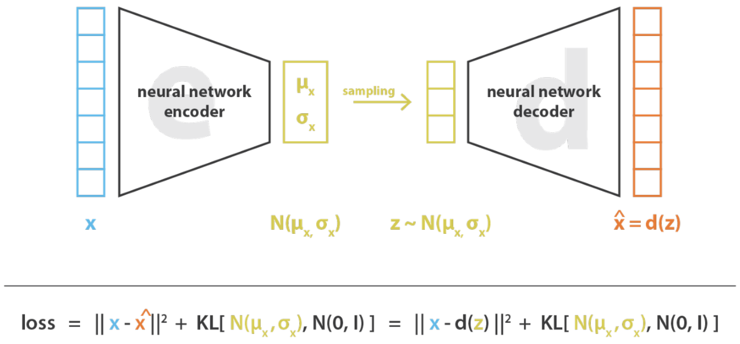

neural network encoder
μ_x
σ_x
sampling
neural network decoder
x
$N(\mu_x, \sigma_x)$
$z \sim N(\mu_x, \sigma_x)$
$\hat{x} = d(z)$
$\text{loss} = \| x - \hat{x} \|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = \| x - d(z) \|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$

Figure 3: Variational Auto-Encoder Architecture

3 Data

GTZAN dataset: We used the GTZAN dataset which is cited in both papers [1] and [2]. This dataset consists of 1000 music time series data (of 30 seconds) distributed in 10 music genre (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock). The classes are balanced out.

DBN: For the data preprocessing of the DBN we divided the audio into short frames of 46.44ms (1024 samples at 22050 Hz sampling rate) and compute the absolute values of their Discrete Fourier Transform (DFT), and considering symmetry in the DFT, we ended up with inputs of dimension 513. This is the input we have given to train our DBN constituted of 3 RBMs with 50 units per layer.

convRBM & convVAE: The samples were preprocessed before being sent to the convolutional networks. We used Mel spectrograms as "image" representations of the signals. Mel's spectrograms is more robust for music processing. It mimics how humans perceive frequency: finer resolution at lower frequencies and coarser resolution at higher frequencies. The length of the window was 1024 with 50% overlap.

4 Results

Training deep learning networks turned out to be a more challenging task than we thought in the beginning. We had unsuccessful attempt at training a ConvRBM, subpar results with VAE and solid classification performance with DBN

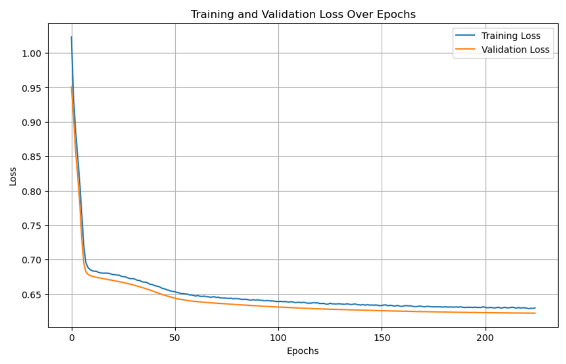
DBN: For the Deep Belief Network with the final "fine-tuning" with an added final layer, we achieved to obtain a classification accuracy of 59% by pretraining each RBM for 5 epochs and doing the fine-tuning on 18 epochs. Our DBN was composed of 3 RBMs with 50 units per layers. To compare our result with the results from the paper [1], they obtained a classification accuracy of 73% for 473 supervised epochs (we stopped at 18 after 2 hours of training).

ConvRBM: Although suggested for the problem of the analysis of music by the original paper [2], it performed rather poorly in our case. We experimented with two different set-ups - using the 64 and 96 frequency bins Mel spectrograms of the audio samples. We trained ConvRBM for 325 and 140 epochs, respectively. The loss plots are in Figure 4 in the Appendix. For both cases, we used the first 10 seconds of the audio extract to reduce the computational complexity. We attempted several pooling strategies after the last layer and before training the classifiers, but results ended up being poor. The highest $F1$ score achieved is 52%. The classifier used after ConvRBM is a Random Forest. We plotted the t-SNE graphs for the features for the global max pooling approach, Figure 5 and 6 available in Appendix, shedding light onto lack of good separation in latent space from the learned features. The complete accuracy testing results are available in Table 1 for the case of 64 bins and 2 for 96 bins.

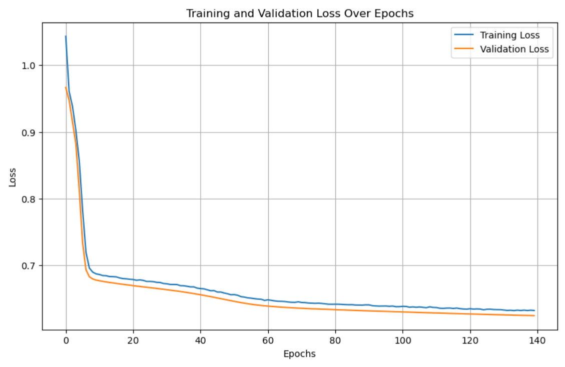
VAE: For the convolutional VAE, we used the spectrogram prepossessing explained before in the data section. As the input is an image, we used convolutional layers for the encoder and decoder part. After training our VAE for 100 epochs, we used the trained VAE model to encode our data which represented the features, we pass them to a logistic regression model, to solve the classifcation task and we obtained a classification accuracy of 38%. Moreover if the latent space is a good representation of the input space, we can sample a new data, decode it and it will generate a new audio data!

Appendix

.1 Training and validation loss for ConvRBM



(a) Loss with 64 bins



(b) Loss with 96 bins

Figure 4: Plots of Training and Validation Losses during training of ConvRBM.

.2 Visualization of the latent space for ConvRBM

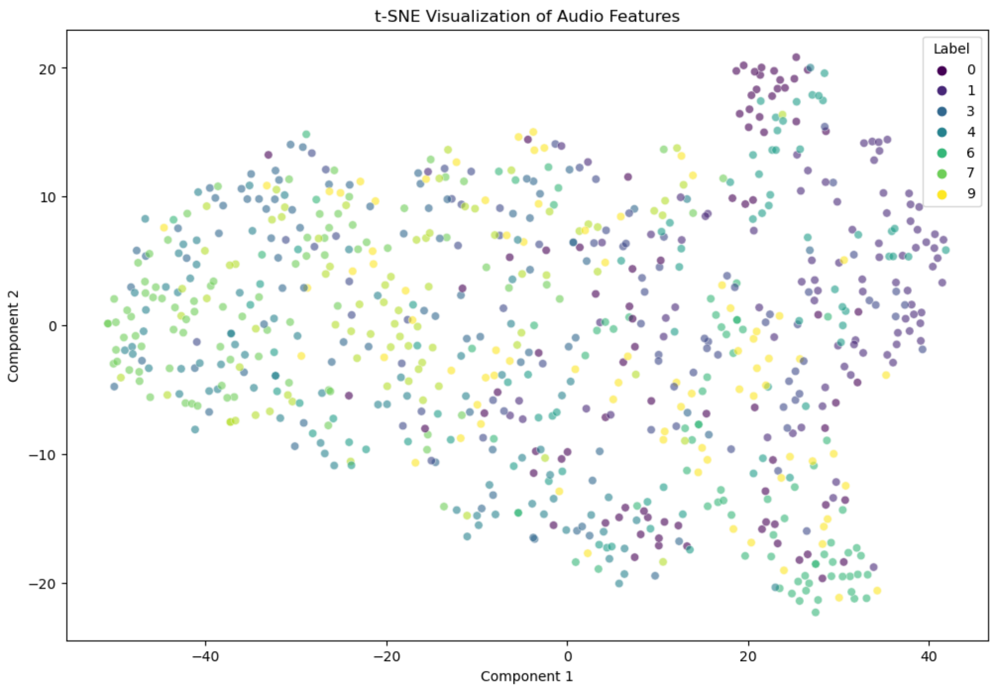


Figure 5: t-SNE visualization of the features extracted global max pooling for ConvRBM while using 64 bins. Different colors represent different genres.

.3 Accuracy scores for ConvRBM

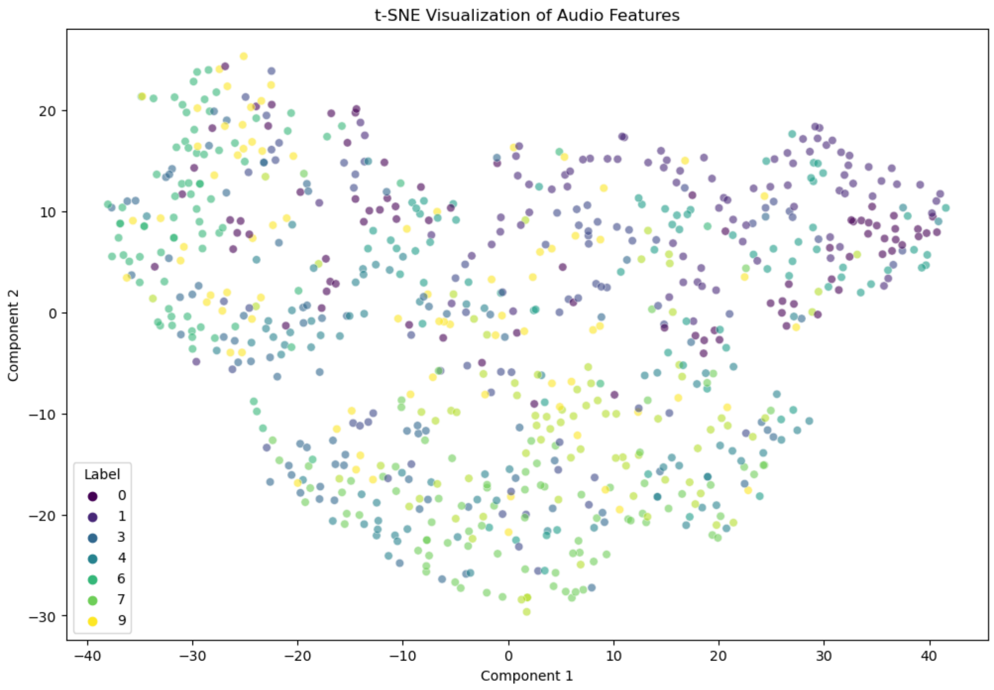


Figure 6: t-SNE visualization of the features extracted global max pooling for ConvRBM while using 96 bins. Different colors represent different genres.

Table 1: Accuracy of the Random Forest classifier on features from ConvRBM for Different Pooling Approaches with 64 bins

Class	Flatten	Global Avg Pooling	Max Pooling	Global Max Pooling	Avg Pool2D
0	0.33	0.33	0.33	0.60	0.29
1	0.36	0.09	0.56	0.67	0.42
2	0.50	0.25	0.50	0.50	0.25
3	0.20	0.33	0.12	0.42	0.00
4	0.56	0.22	0.22	0.25	0.29
5	0.88	0.90	0.78	0.67	0.83
6	0.76	0.80	0.67	0.75	0.79
7	0.29	0.40	0.36	0.33	0.36
8	0.36	0.25	0.33	0.43	0.33
9	0.00	0.25	0.33	0.00	0.17
Overall	0.46	0.41	0.44	0.50	0.39

References

[1] Philippe Hamel and Douglas Eck. LEARNING FEATURES FROM MUSIC AUDIO WITH DEEP BELIEF NETWORKS, 2010.

[2] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42, 2014.

[3] Mohammad Norouzi, Mani Ranjbar, and Greg Mori. Stacks of convolutional restricted boltz-

Table 2: F1-Score for Different Pooling Approaches with 96 bins

Class	Flatten	Global Avg Pooling	Max Pooling	Global Max Pooling	Avg Pool2D
0	0.47	0.20	0.33	0.53	0.50
1	0.59	0.19	0.59	0.36	0.71
2	0.47	0.24	0.27	0.37	0.50
3	0.29	0.25	0.29	0.37	0.17
4	0.32	0.22	0.22	0.56	0.22
5	0.67	0.38	0.50	0.58	0.70
6	0.75	0.75	0.80	0.80	0.82
7	0.40	0.43	0.42	0.38	0.40
8	0.29	0.38	0.40	0.52	0.24
9	0.21	0.13	0.22	0.38	0.36
Overall	0.47	0.35	0.44	0.52	0.50

mann machines for shift-invariant feature learning. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2735–2742. IEEE, 2009.