




# LEARNING SYMBOLIC EMBEDDING SPACES FOR MUSICAL COMPOSITION AND INFERENCE

MATHIEU PRANG

*Supervised by*

Pr, Philippe Esling


Master's Degree
Music Representations team
IRCAM / UPMC / Télécom ParisTech
February - August 2017

Computers are getting smarter all the time.
Scientists tell us that soon they will be able to talk to us.
And by 'they', I mean 'computers'. I doubt scientists will ever be able to talk to us.

— Dave Barry

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald Ervin Knuth

## ABSTRACT

This internship aims to develop new representations of musical symbolic. Whereas the previous approaches are based on known mathematical rules, we tried to develop a more empirical model through machine learning framework. Our goal is to represent musical symbols in a space that carry semantics relationships between them, called *embedding space*. This approach allows to extract new descriptive dimensions that may be relevant for music analysis and generation. Previous work in Natural Language Processing provided such very efficient representation of words. Hence, starting from the possible analogy that exists between a text and a musical score, we tried to adapt the best state-of-the-art word embedding algorithms to musical data. However, some critical differences between the structural properties of musical and textual symbols, prompted us to develop a machine learning model especially tailored for music. To that end, we used a Convolutional Neural Network in order to capture visual features of the piano-roll representation of chords independently of the pitch class. Then, we added a Long Short Term Memory network that aim to integrate information about time dependencies of musical object in the final embedding space. Finally we proposed some applications that allow to infer knowledge on musical concepts or to encourage musical creativity.

## RÉSUMÉ

Ce stage a pour but de développer de nouvelles représentations de la musique symbolique. Alors que les précédentes approches sont basées sur des règles de mathématiques prédéfinies, nous avons essayé de développer un modèle plus empirique grâce au système d'apprentissage machine. Notre but est d'apprendre comment représenter les symboles musicaux dans un espace, appelé *espace d'embedding*, où apparaissent les relations sémantiques que partagent ces objets. Cette approche permet d'extraire de nouvelles dimensions descriptives pouvant plus tard être utile à l'analyse et la génération de musique. De précédentes études dans le domaine du traitement automatique du langage naturel ont conduis à des espaces d'embedding très efficaces pour la représentation des mots. Ainsi, partant de la potentielle analogie qui existe entre un texte et une partition de musique, nous avons essayé d'adapter les meilleurs modèles de l'état de l'art des embeddings pour les mots aux symboles musicaux. Cependant, il existe des différences majeurs entre la structure d'un texte et celle d'une musique qui nous a poussés a développer un modèle particulier pour la musique. Pour cela, nous avons utilisés un réseau de neurones convolutionnels pour capturer les caractéristiques harmoniques, indépendamment de l'octave, que forme la représentation en piano-roll des accords. Puis nous avons exploré l'utilité de l'ajout d'un réseau LSTM qui a pour objectif d'intégrer des informations sur la dépendance temporelle des symboles musicaux dans l'espace d'embedding. Finalement, nous avons proposés des applications qui nous permettent d'inférer des connaissances sur les concepts musicaux et d'aider la créativité musicale.

# ACKNOWLEDGMENTS

First and foremost I want to thank my supervisor Philippe Esling for his remarkable commitment in this internship. His friendly guidance and expert advice have made this work profoundly pleasant and captivating. Besides the enthusiastic vision he has for a wide variety of things, I could benefit from his vast scientific knowledge that allowed me to improve my competences in many fields. I am also very thankful for his precious support concerning the further development of my professional life.

My sincere thanks also goes to all future and already seasoned researchers at IRCAM who helped me through their insightful comments. Especially Leopold Crestel who showed genuine interest in my work and contributed largely to it with many brilliant ideas.

I am indebted to Nicolas Obin who provided to me a valuable help for integrating this stimulating master degree. I take this opportunity to express my sincere gratitude to all the actors of ATIAM course and particularly to all my student colleagues. with whom i spend a tremendous year. I would like to thank my friend Hadrien Foroughmand who provided me unfailing support and so many unforgettable moments during these intense years of higher eduction.

Last but not the least, a special thanks to my family for their love and incredible support. For my parents who raised me with a love of music and sciences and encouraged me in all my pursuits. For my two brothers whose wise guidance has given so much to me over the years. And most of all, for my loving, supportive, and awesome Camille who give me the strength to overcome difficulties and keep the faith on me. Thank you.

# CONTENTS

# 1

# INTRODUCTION

When we hear an orchestral piece or read its musical score, we implicitly interpret sets of information inside this complex data. Indeed, we can perform a link between this information and previously known concepts (e.g. the piece is sad, percussive, melancholic, played in a concert hall or in the street). These concepts can be thought of as *high-level abstractions*, oppositely to low-level ones like acoustic signal values.
In the past decades, the field of *computer music* has precisely targeted these problems of understanding musical concepts. Indeed, it is with this information that we can provide tools to help composers and listeners but also define methods of analysis and composition that improve our musical knowledge. Nowadays, a wide variety of approaches have been taken forward but by doing a comparison, we can see that there all largely rely on one crucial point: *the way we represent music*.
This question has stirred up a huge interest in the music researchers community that leaded on a lot of very interesting and efficient representations referred to *musical spaces*. If these formalizations efficiency was acknowledged, there all share the same development process that consist of building the space through known mathematical rules before representing any musical data. In other words, these spaces are human-designed and based on existing knowledge. But could we try to let the computer itself learn an appropriate representation of music ?
In order to allow a machine to understand these concepts and disentangle the correlations that exist in music, this machine should be given a way to interpret orchestral scores as humans do. However, even if we manage to gather a wide set of musical information, constructing techniques to understand music in previously unseen contexts seems like a daunting task [9]. Tackling these difficult questions is the goal of the *machine learning* field.
In this chapter, we introduce the goals and context of this internship and the general principles behind machine learning. We explain how deep learning can be a promising framework to study these complex tasks and introduce the different approaches and models which will be used as baselines for our work.

## 1.1 OBJECTIVES

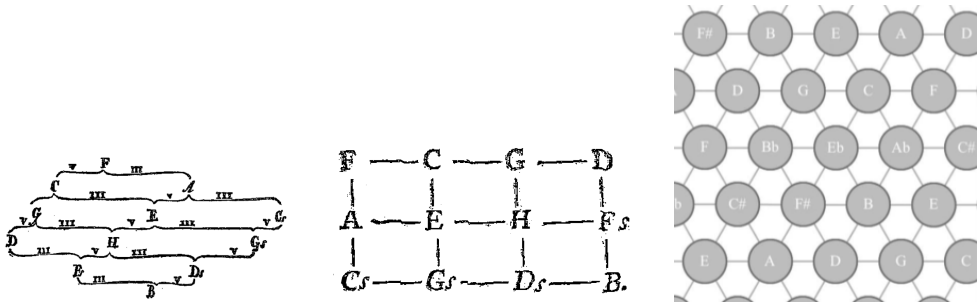
The main objective of this internship is to tackle the problem of finding efficient spaces for representing music. Indeed, over the several compositional tools have been developed over the past years [2, 4], the critical aspects in their functioning is the musical representation itself. We provide here an overview of the musical spaces and computer-assisted composition tools. Then, we introduce our assumptions and hypotheses regarding the use of learning algorithms for these kind of tasks.

### 1.1.1 *Music as symbols*

Music has been transcribed in a written format from a very ancient times (with elements of musical scores seeming to date back to 1400 BC. [13]. The marks and symbols that were developed along the past millenniums gradually informed on the duration and pitch of the corresponding melody. Then dynamic and instrumentation of the different notes were introduced based on the ever-expanding human instrumentarium [3].*This evolution has shaped musical notation as we know it today*. Hence, we can see that the representation of music as symbols itself has been a central question in the history of music. By increasingly specifying the notation, musicians could play a piece of music closer to the original composer's intention. In that sense, musical notation could be thought of as a model which enables us to reason and think about music.
Nowadays, with the apparition of the digital era, multiple machine-readable scores formats have been developed such as the Musical Instrument Digital Interface (MIDI). This type of digital data format allows to treat music through its symbolic representation since it is based on a finite alphabet of symbols. A MIDI file encodes information for the different notes, durations and intensity through numerical messages with a pre-defined temporal quantification (a subdivision of a quarter note). Hence, this format has been widely used in computer music research as it allows a compact representation of music. Other formats have been developed using for instance LISP [4] or XML [22]. However, in our internship, we will rely on the piano-roll format extracted from MIDI files as this allowed us to construct a large database of music.

### 1.1.2 *Musical spaces*

One of the core research question in computer music remains to find an adequate representation for the *relationships* between musical objects. Indeed, the transcription of music as symbols usually fails to provide information about harmonic or timbral relationships. In that sense, we can loosely say that the




(a) First version of Euler's Tonnetz in 1739.

(b) Another version of Euler's tonnetz in 1774.

(c) IRCAM tonnetz used in the Hexachord software.

Figure 1: Three different representation of Tonnetz. The tonal space are represented in a grid where we transcribe the harmonic progression of a musical piece. Figure 1a from [20], Figure 1b from [21], Figure 1c from [10]

goal would be to find a target *space*, which could exhibit such properties between musical entities. Hence, finding representations of musical objects as spaces has witnessed a flourishing interest in the scientists community [11, 65–67]. Many of the formalizations proposed over the past decades entail an algebraic nature that could allow to study combinatorial properties and classify musical structures. Here, we delimit a distinction between these methods into two types of representations: the *rule-based* and the *agnostic* approaches.

In the rule-based stream of research, several types of spaces have been developed since the Pythagoreans. Indeed, Marin Mersenne allowed to discover many algrebraic and geometric structure in classical music through his circular representation of the pitch space in the 17th century [43, 44, 68]. Many years later Henry Klumpenhouwer present a new space for representing music called the K-nets [34]. This approach leaded to reveal some structural aspects in music through the many isographies of the networks [39, 40, 53]. Finally, we can cite the well-known Tonnetz, a musical space invented by Euler in the 18th century [20]. The main idea behind it is to represent the tonal space in a grid (as depicted in figure 1) and then used it to put forward harmonics relationships in musical pieces.

There are two main benefits of this type of rule-based approach. First, once the model is built, it can be straightforward to analyze some of its properties (based on the defined sets of rules). Second, we can also understand the scope where the model should be efficient based on its construction.

But as it is defined, a rule-based approach represent the particular vision of the designer that has thought and crafted the corresponding rule sets. Hence, the corresponding musical spaces will provide a given set of interactions. It is interesting to ask if we could develop a more empirical discovery of these spaces that could provide more generic musical relationships. These kinds of spaces could allow to exhibit properties in musical scores in a way

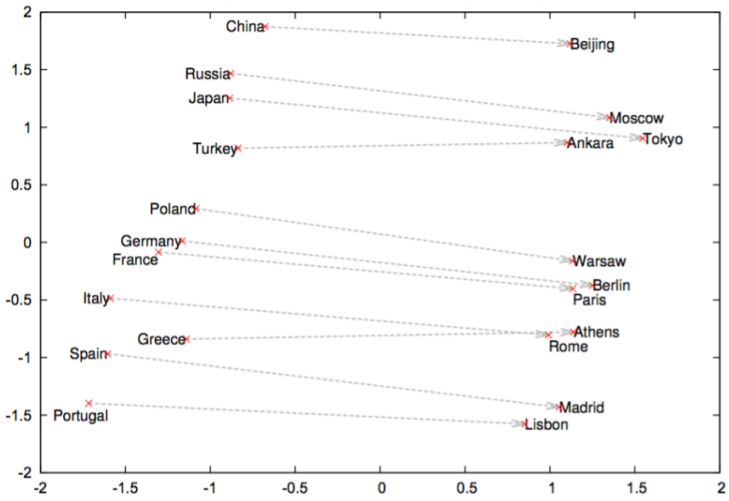



Figure 2: Country and capital embedding vectors projected by PCA. The figure shows the possible ability of an embedding space to capture information about human concepts and the relationships between them. Figure from [46]

that we never would have thought. In doing so, we could then find some new relevant features and metric relationships between musical entities and develop innovative applications. Hence, in the following, we consider that the important properties of a space are not necessarily its *dimensions* (like in the rule-based approaches), but rather the *metric relationships* or *distances* between objects inside this space (like in the agnostic approach that we seek to develop) [46].

However, we remain conscious of the limitations of such agnostic spaces. Indeed, these are still indirectly the product of our design of the learning algorithms. Furthermore, they might be highly dependent on the dataset (see 2.1) used for their construction. Finally, there might be no direct ways to analyze their properties nor prove their efficiency.

### 1.1.3 *Learning spaces*

Recently, different breakthroughs in machine learning (and most notably in the Natural Language Processing (NLP) field), has provided steps towards our over-arching goal. This branch of computer science traces back to the 1950s and have seen some major progress during the last decade. Particularly relevant to our work is the huge step forward in the development of word *embedding spaces*. In these approaches, large datasets of sentences are used to understand the relationships between words. The goal is to find a space where words are represented as points (vectors), whose distances in the space mirrors the semantic similarity between words. We can see an example of this kind of space projected by PCA in Figure 2. We can see that

the distance between all the countries and their corresponding capitals are almost the same. It illustrates the ability of this space to capture information about concepts and the relationships between them [46].
By using such vectors as a representational basis for other machine learning tasks, scientists made colossal improvements and opened a lot of possibilities for a wide variety of powerful applications. For instance, Tang et al. developed a tool that classify the messages from Twitter according to their sentiments [63]. Palangi et al. used word embedding to perform document retrieval or web search tasks [50].

In our context some structural aspects in both field, language and music, could hypothetically share some logical equivalence. Indeed, a sentence is composed by words hierarchically located as a melody is composed by notes. Moreover this kind of learning space could also be very valuable for the musical analysis and composition field. As well as being a potential analysis and knowledge inference tools itself it could be an efficient and new basis representation of music for many creative application. Moreover, this continuous space could provide melody generation or transformation directly from it. But one of the most interesting challenge will be to link these informations with perception or signal processing knowledge as it has be done in other field [5, 31, 32, 48]. Armed with this combined space we could find some relevant features about music and develop powerful classification or recommendation tools.
Setting out from this premise we decided to work on this kind of spaces for musical symbolic.

# 2

# STATE-OF-THE-ART

## 2.1 PRINCIPLES OF MACHINE LEARNING

### 2.1.1 *Formal description*

Learning can be defined as the process of acquiring, modifying or reinforcing knowledge by discovering new facts and theories through observations [23]. To succeed, learning algorithms need to grasp the generic properties of different types of objects by "observing" a large amount of examples. These observations are collected inside training datasets that supposedly contain a wide variety of examples.
There exists three major types of learning :

- ***Supervised learning***: Inferring a function from labeled training data. Every sample in the dataset is provided with a corresponding groundtruth label.
- ***Unsupervised learning***: Trying to find hidden structure in unlabeled data. This leads to the important difference with supervised learning that correct input/output pairs are never presented. Moreover, there is no simple evaluation of the models accuracy.
- ***Reinforcement learning***: Acting to maximize a notion of cumulative reward. This type of learning was inspired by behaviorist psychology. The model receives a positive reward if it outputs the right object and a negative one in the opposite.

In computer science, an example of a typical problem is to learn how to classify elements by observing a set of labeled examples. Hence, the final goal is to be able to find the class memberships of given objects (e.g. a sound played either by a piano, an oboe or a violin). Mathematically we can define the classification learning problem as follows.
Given a training dataset $\chi$ of $N$ samples $X = \{x_1, ..., x_N\}$ with $x_N \in \mathbb{R}^D$, we want assign to each sample a class inside the set $Y = \{y_1, ..., y_M\}$ of $M$ classes with $y_M \in \{1, ..., M\}$.

To do so, we need to define a model $\Gamma_\theta$ that depends on the set of parameters $\theta \in \Theta$. This model can be seen as a *transform* mapping an input to a class value such that

$$\tilde{y} = \Gamma_\theta(x_i) \tag{1}$$

In order to define the success of the algorithm, but also to allow learning, we further need to define a ***score function***,

$$h = h_\theta : \mathbb{R}^D \to \mathbb{R}^M \tag{2}$$

and a ***loss function***,

$$\mathcal{L} = \mathcal{L}_{X,Y} : \Theta \to \mathbb{R} \tag{3}$$

The role of the score function is to determine the membership of a given sample $x_i$ to a class $y_m$. In a statistical setting, we can interpret this function as the probability of belonging to a given class.

$$h_m(x_i) = p(\tilde{y} = y_m \mid x_i, \theta) \tag{4}$$

Considering the basic rules of probability distributions, we have

$$\begin{cases} \sum_{m=1}^{M} h_m(x) = 1 \\ \\ h_m(x) \in [0,1], \qquad\qquad \forall x \in \chi, \quad \forall m = 1, ..., M \end{cases} \tag{5}$$

On the other hand, the loss function computes the difference between predictions of the model and groundtruths. In order to learn the most efficient model, we need to update its parameters $\theta$ by minimizing the value of this loss function

$$\hat{\theta} = \mathrm{argmin}_{\theta \in \Theta}(\mathcal{L}_{X,Y}) \tag{6}$$

There is usually no analytical solution and sometimes not even a single minimum for this problem. Therefore, we rely on the *gradient descent* algorithm [15] to find a potential solution (minimum) to this problem by updating the parameters iteratively depending on the gradient of the error

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}_{X,Y}(\theta_n) \tag{7}$$

Where $\eta$ is the *learning rate*, which defines the magnitude of the parameters update in the direction of the errors gradient as depicted in Figure 3.

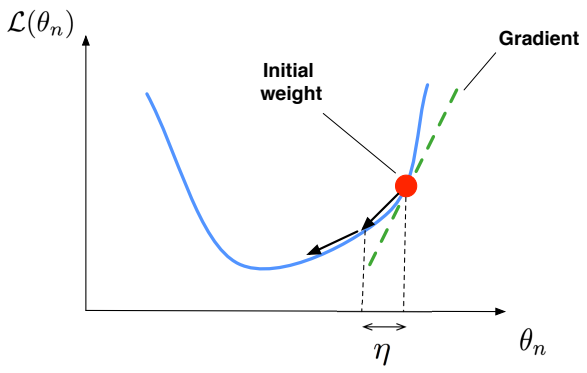



Figure 3: Gradient descent on the loss function $\mathcal{L}_{X,Y}(\theta_n)$. The learning rate $\eta$ corresponds to the magintude of the update of the parameters $\theta$.

Obviously, the convergence towards a global minimum depends on the learning rate but also on the properties of the loss function. Indeed, $\mathcal{L}$ should preferably be convex, which it is not always the case, and the existence of several local minima can make the training really difficult.

### 2.1.2 *Training*

A single phase of training is referred to an *epoch*, which corresponds to one iteration of the training loop defined as listed in Algorithm 1.

**Input** : $\mathcal{L}_{X,Y}(\theta_n)$, X, Y
**Result** : The parameters $\theta$ of the model $\Gamma_\theta$ minimize the value of the loss function $\mathcal{L}_{X,Y}(\theta_n)$.
Random intialization of $\theta$;
**while** $\hat{\theta} \neq \operatorname{argmin}_{\theta\in\Theta}(\mathcal{L}_{X,Y})$ **do**
  *Compute a prediction* $\tilde{y}$ *for inputs* $x_i$ *depending on the current parameters* $\theta_n$
  $\tilde{y} = \Gamma_\theta(x_i)$;
  *Evaluate the error by comparing the predictions groundtruth* $y_i$ *with an arbitrary distance function* F
  $\mathcal{L}_{X,Y} = F(\tilde{y}, y_i)$ ;
  *Update the parameters of the model in order to decrease the loss value, by relying on the derivatives of the error*
  $\theta_{n+1} = \theta_n - \eta\nabla_\theta\mathcal{L}_{X,Y}(\theta_n)$ ;
**end**

**Algorithmus 1 :** Training algorithm of a model $\Gamma_\theta$ for a classification task. X is the training set composed by a N samples $X = \{x_1, ..., x_N\}$, Y is the set of M classes that correspond with the samples $Y = \{y_1, ..., y_N\}$ with $y_i \in \{1, ..., M\}$

At this point, the question arises to know whether the number of epochs has to be the largest possible to have the most efficient model. Unfortunately, this is generally not the case due to the phenomenon known as *over-fitting*. Indeed, if the model learns "too precisely" the training dataset properties, it will learn the samples themselves and will not be able to generalize its learned concepts to unseen contexts anymore. This can be understood graphically by looking at Figure 4.

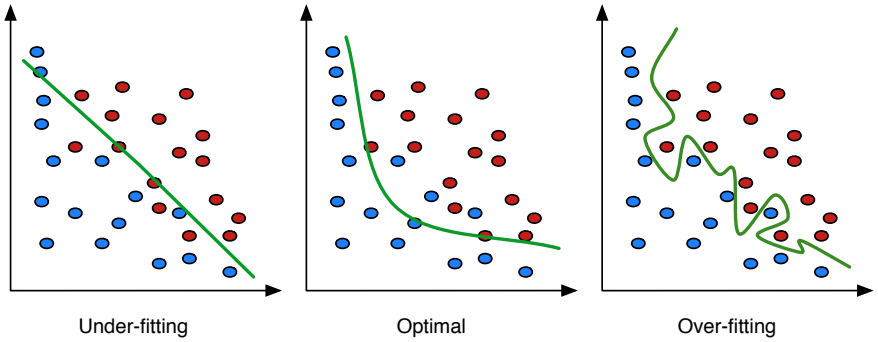



Figure 4: Classification task defined by the boundaries of the model in the case of under-fitting, optimal training and over-fitting.

The first case on the right illustrate a model that has not been trained enough. Its classification function is too simple to seperate efficiently the sample. In the opposite we can see on the right a model that has made too much training epoch. In this case, it will be very efficient for tasks on this particular dataset but very bad on other example. Finally, the center of the figure shows a model that has made the optimal number of epochs. To prevent the over-fitting, we split the data into three datasets, *training set*, *testing set*, and *validation set*.
We use the training dataset in order to update the parameters, and then we compute the loss with the test one. When the training loss value tends to zero (a model "knows" each sample, zero miss-predictions are made), the test loss will re-increase because of over-fitting as depicted in Figure 5. We stop the training at this point and assess our model on the validation dataset to get the expected final accuracy of the model on unknown data.
Another solution to prevent over-fitting is to apply *regularization* to the model or the learning process. Examples of regularization include adding noise to the input or restrict the values of the parameters through weight decay. Regularization allows preventing the model to learn too precisely the training samples.

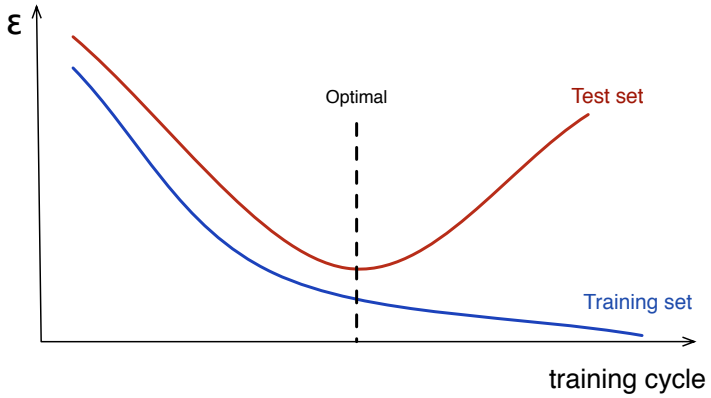



Figure 5: Loss value with the training set and the test set. Optimal number of epoch is reach when the red curve begin to grow up again.

## 2.2 DEEP LEARNING

*"When a function can be compactly represented by a deep architecture, it might need a very large architecture to be represented by an insufficiently deep one."*
*- Y. Bengio*

Coming back to our original problem, we discussed in Chapter 1 that we seek an algorithm able to link low-level abstractions with high-level ones. However, if we depict the problem of trying to find the name of a piece based on its musical score we can see that we need to pass through many gradually abstraction levels as shown in figure 6.

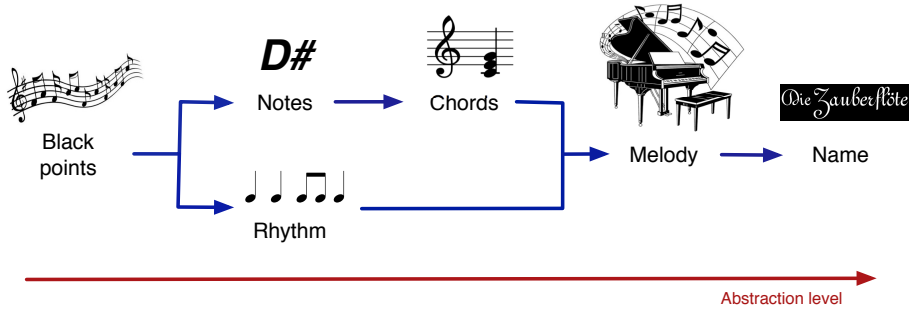



Figure 6: Decompsiting the problem of retrieving the name of a song to multiple abstraction level.

Some mammals can usually solved this type of complex problem with abstractions hierarchy because their brain have a deep architecture where each level correspond to different areas of the cortex. Inspired by these observations, the deep learning approach appeared in the machine learning community.
However, the major issue with multi-layer models stems from *the gradient diffusion problem*. As we saw it before, the parameters update is proportional to the loss function gradient (Equation 7) which is in a range of $[-1, 1]$. Therefore, with a given N layers depth architecture this has the effect of multiplying N times this small numbers in order to update the output layer weights.

that is why the gradient magnitude gradually vanishes and the training step will decrease exponentially with the last layers training very slowly [26].
However, in 2006, Hinton et al. found a way to avoid this issue. Their algorithm called *greedy layer-wise learning* allows each layer of the network to be trained independently and in an unsupervised manner [24]. It is defined as follow

1. Learn weights of the first layer assuming all the other weights are tied. (We can see it as the higher layers do not exist but for an approximation to work well, we assume that they have tied weights.)
2. Freeze this weights and use it to infer factorial approximate posterior distributions over the states of the variables in the first hidden layer, even if subsequent changes in higher level weights mean that this inference method is no longer correct.
3. Learn a model of the higher level "data" untying all the higher weights matrices from the first one.

By applying this approach repeatedly from the first layer to the last, we learn a deep, densely-connected network one layer at a time.
We now present some very common and effective deep neural networks which were useful for our work.

### 2.2.1 *Neural Networks*

Now that we have a global overview of learning algorithms, we will define the model $\Gamma_\theta$ as a neural networks. A wide part of research in machine learning has focused on the concept of artificial neural networks. Indeed, the most widely known basis of intelligent behavior as we know it, is the biological neuron. Therefore, scientists tried to mimic its mechanisms, tracing back to the original model of McCulloch and Pitts in 1943 [42]. An artificial neuron is composed of multiple weights and a threshold, that together define its parameters, as depicted in Figure 7. To decide if the neuron will activate its output or not, it uses an affine transform and a non-linear activation function.
Mathematically, with N inputs a neuron output is defined as

$$y = \sum_{i=1}^{N} x_i w_i + T \tag{8}$$

with $w_i$ the learned weights and T the threshold of activation. If we interpret this equation geometrically, we can see that it corresponds to an N-dimensional hyperplane. Therefore, a neuron can divide a space (akin to binary classification), or approximate a function (as the sum will give an output whatever X comes in). By organizing these neurons as layers where

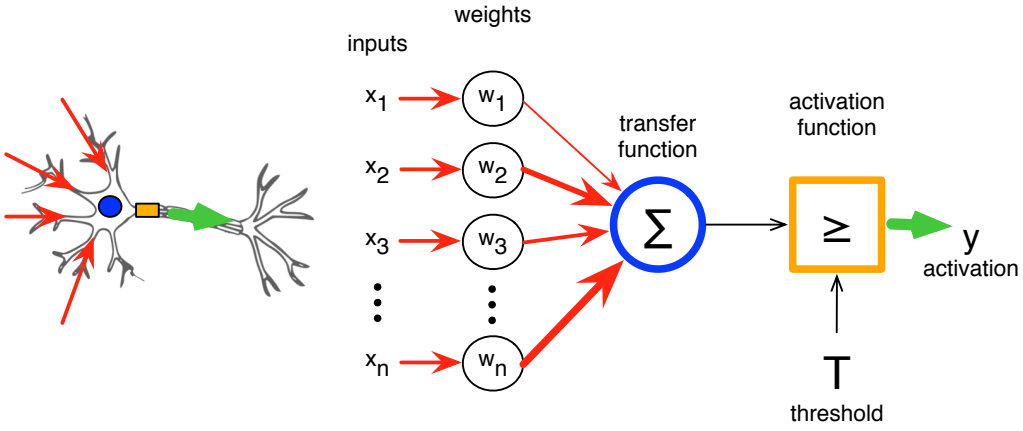
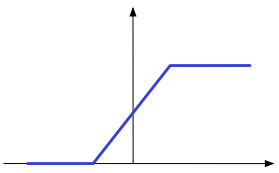



Figure 7: A biological neuron (left) is approximated through affine transform and activation function (right).

each neuron (also called units) transforms the input independently, we obtain the *perceptron*. These layers are then stacked one after another, where the input of a layer is the output of the previous one, to obtain the well-known multi-layer perceptron. The number of layers is call the *depth* of the model. Hence, the multi-layer neural network allows combining non-linear activations in order to process more complex tasks.
However, a problem arise from this representation. Indeed, to update the parameters with gradient descent method (Equation 7) the output space has to be continuous (e. g., the value of $y$ must be continuous to be differentiable, $y(x)$ is of class $\mathcal{C}^1$). To alleviate this problem we need to use different activation function denoted as $\tau(x)$, and we now consider $T$ as a bias denoted as $b$, leading to $y = \tau(\sum w_i x_i + b)$. Three examples of common activation functions are showed in the following.

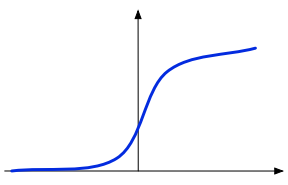
- Piecewise linear :

$$\tau(x) = \begin{cases} 0 & \forall\ x \leqslant x_{min} \\ mx + b & \forall\ x_{max} > x > x_{min} \\ 1 & \forall\ x \geqslant x_{max} \end{cases}$$



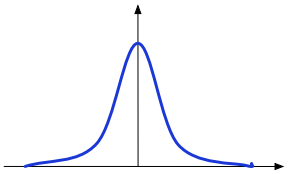
- Sigmoid :

$$\tau(x) = \frac{e^x}{1 + e^{-\beta x}}$$



- Gaussian :

$$\tau(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



We illustrate one of the most common architecture in figure 8, the fully-connected network whose units between two adjacent layers are fully pairwise connected. This type of architecture is called *feed-forward* as the information moves in only one direction, forward, from the inputs to the outputs. There is no loops, backward connections or connections among units in the same layer. Moreover, the middle layers have no link with the external word, and hence are called *hidden layers* [38].

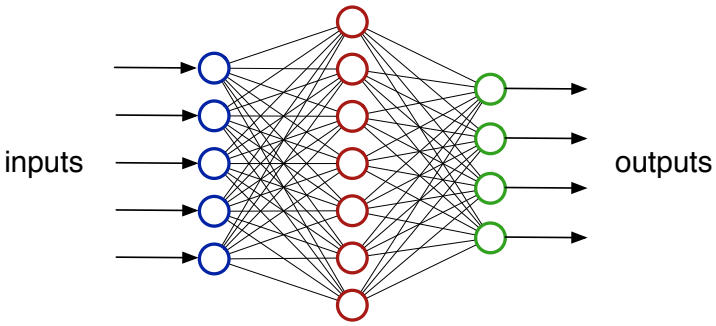



Figure 8: A fully-connected network with three layers. Units between two adjacent layers are fully pairwise connected.

Other type of network based on different connections or operations have been developed. We now present two of this different architectures that were usefull for our work, Convolutional Neural Networks and Long Short Term Memory network.

### 2.2.2 *Convolutional Neural Networks*

Convolutional Neural Networks (CNNs) were inspired by the models of the visual system's structure proposed by Hubel and Wiesel in 1962 [28]. It is a category of neural networks that have proven very effective in areas such as image recognition and classification [35, 58, 59]. In audio, we can use CNNs with inputs which outputs a 2-dimensional matrix such as the Short-Term Fourier Transform [37]. There are four main operations in this kind of network, each processed by a different layer (see Figure 9).

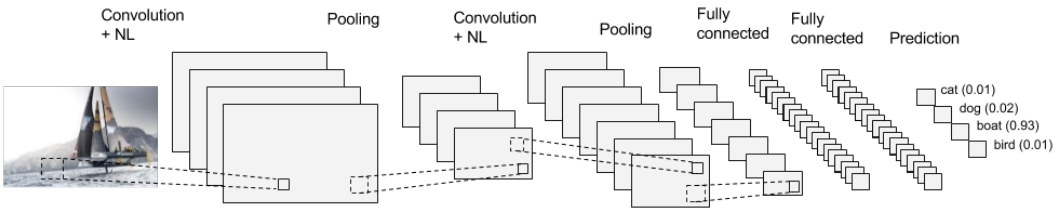



Figure 9: Convolutional neural network with two convolutional layers each followed by a pooling layer and two fully connected layers for classification.

CONVOLUTION The first layer is the convolution operator. Its primary purpose is to extract features from the input matrix. Indeed, each units $k \in \mathbb{N}$ in this layer can be seen as a small filter determined by the weights $W_k$ and the bias $b_k$ that we convolve across the width and height of the input data $x$. Hence, this layer will produce a 2-dimensional activation map $h^k$, that gives the activation of that filter across every spatial position

$$h^k_{ij} = (W^k * x)_{ij} + b_k \tag{9}$$

With the discrete convolution for a 2D signal defined as

$$f[m,n] * g[m,n] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u,v]g[m-u,n-v] \tag{10}$$

The responses across different regions of space are called the *receptive fields*. During the training process, the network will need to learn filters that can activate when they see some recurring features such as edges or other simple shapes. By stacking convolutional layers, the features in the upper layers can be considered as higher-level abstraction such as composed shapes.

NON-LINEARITY As discussed in the previous section, we have to introduce non-linearities (NL) in the network in order to model complex relationships. Hence, before stacking every feature maps in order to obtain the output activations we apply a non-linear function like those introduced previously or the recently proposed ReLU (Rectified Linear Unit) defined by $\mathrm{output} = \max(0, \mathrm{input})$ [18].

POOLING OR SUBSAMPLING Spatial pooling (also called subsampling or downsampling) allows to reduce the dimensionality of each feature map. The principle behind the pooling operation is to define a spatial neighborhood (such as a $3 \times 3$ window) and take the largest elements (*max-pooling*) or the average (*average-pooling*) of all elements in that window. In that way, we progressively reduce the spatial size of the input representation and make it more manageable.

CLASSIFICATION Based on the highest level features in the network, we can use these to classify the input into various categories. One of the simplest way to do that is to add several fully-connected layers (Figure 8). By relying on this architecture, the CNN will take into account the combinations of features similarly to the multi-layer perceptron.

### 2.2.3 *Long Short Term Memory (LSTM)*

Traditional neural networks do not allow information to persist in time. In other words, it could not use its reasoning about previous events to inform later ones. To address this issue, specific networks with loop connections have been developed, called *Recurrent Neural Networks* (RNN) [19]. The idea is that a neuron now contains a loop to itself, allowing to carry information from one time step to the next. Hence, its activation can be defined as

$$h_t = \sigma\left(\sum w_t x_t + h_{t-1}\right) \tag{11}$$

With $\sigma$ a non-linear function, $x_t$ the input, $W_t$ the weights of the neuron and $h_{t-1}$ the activation of the previous time step. However this is a non-causal function that disallows this formalization to be implemented. This issue can be solved by "unfolding" the networks. In other words, these networks can be thought of as multiple copies of the same feedforward network, each passing a message to its successor (see figure 10).

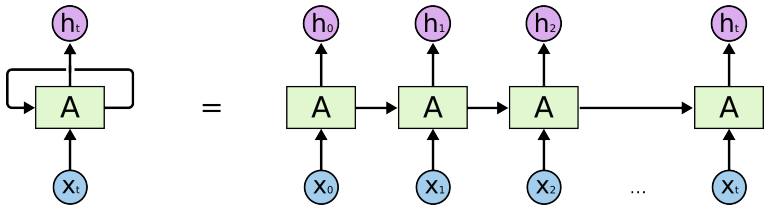
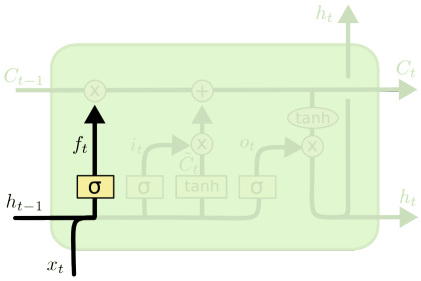



Figure 10: Recurrent Neural Networks can be "unfolded" and then thought of as multiple copies of the same network, each passing a message to its successor (Image from [16]

This chain-like nature underlines the intimate connection that RNN share to sequential events. Unfortunately, this kind of network is usually not able to learn long-term dependencies and are usually bound to succeed in tasks with very short contexts. This problem was explored in depth by Bengio et al. [7], exhibiting the theoretical reasons behind these difficulties, such as the vanishing or exploding gradient.
To alleviate these issues, Hochreiter and Schmidhuber introduced the Long Short-Term Memory network (LSTM) [25]. These cells also have this chain-like structure, but the repeating module has a different structure. Indeed, the key element of a LSTM network is a signal called the *cell state* which runs straight down the entire structure. This signal can be seen as the main

information that is passed from an "unrolled units' to another at each time steps. In the following, we denote the cell state at the time step $t$ as $C_t$. During its crossing, this information can be modified or not or even totally forgotted depending of the input of the network $x_t$ and the output of the previous units $h_{t-1}$. This is the role of four other elements in the structure called *gate* defined with weigths $W$ and bias $b$ that we now present in details.

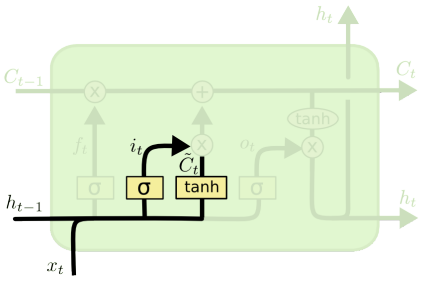
1. ***The forget gate*** is a simple sigmoid layer which is here to decide if we keep the previous information in the cell state or not. It takes into account $h_{t-1}$ and $x_t$, and outputs a number $f_t$ between 0 and 1 where 0 represents "completely forget this" and 1 represents "completely keep this".




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

(Image from [16]).

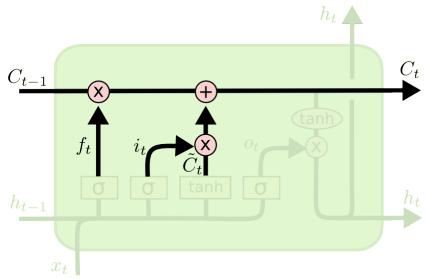
2. ***The input gate*** allows to decide what new information we are going to store in the cell state. This as two parts, a sigmoid layer dedicated to decide which values will be updated or not depending on $i_t$ and a $\tanh$ layer which creates a vector of new candidate values $\tilde{C}_t$ for the update.




$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

(Image from [16]).

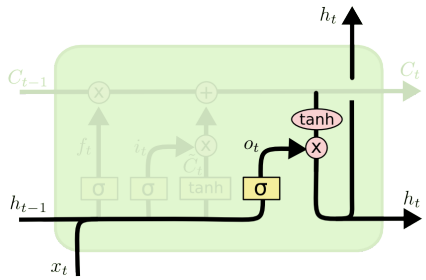
3. ***The update gate*** that its goal is to actually do what we decided before. Therefore we multiply the old state by $f_t$ and add $i_t \times \tilde{C}_t$. The resulting cell state $C_t$ is passed to the next time step unit.




$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

(Image from [16]).

4. ***The output gate*** finally decide what the network is going to output depending of the input $x_t$ and of the cell state $C_t$. The combination of a sigmoid layer and a $\tanh$ layer are applied to achieve this.




$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

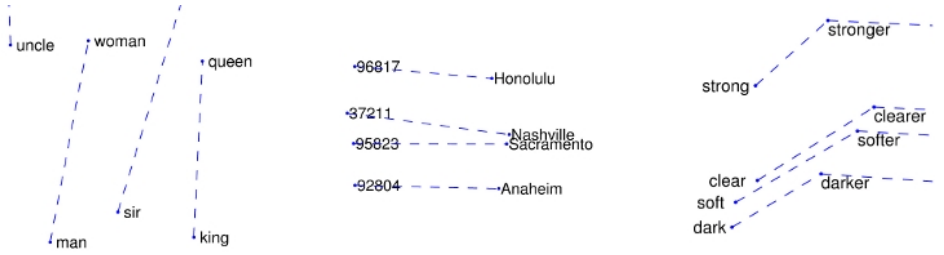
(Image from [16]).

# 3

# RELATED WORK

In this chapter, we develop the previous related work, notably regarding the learning of *embedding spaces*. An embedding space is considered in our context as a space of lower dimensionality which can be found from the high-dimensionality space of the input. Then, inside this space, the embedding of an object will be a representation of this object inside the lower-dimensionality space in such a way that some targeted algebraic properties are preserved. From a topological point of view, one space X is said to be embedded in another space Y when the properties of Y restricted to X are the same as the properties of X. However, in our case, we want to target certain properties of similarities between the different object inputs, such that the distances inside the embedded space mimics the targeted relationships. For our problem, this amounts to find a meaningful representation for musical elements inside a space, in which the distance between the musical entities would faithfully represent their musical similarity. It means that we seek a transform that could map every notes and chords to these low-dimensional vectors, for which the distance between vectors would carry semantic relations. An example of embedding space is depicted in Figure11. For example, in this space, the distance between the word embedding vector for "strong" and the one for "stronger" is the same than between "clear" and "clearer". We can see that even though the dimensions of this space do not have a particular meaning, the metric relationships inside this space do mirror some semantic meaning.

During the last decades, most of the work devoted to embedding spaces has been centered on Natural Language Processing (NLP) trough word embedding space models. Indeed, in 2003 *Bengio et al.* used for the first time a word embedding inside a neural language model [8]. Following this seminal work, many word embedding algorithms were developed including the well-known Latent Semantic Analysis (LSA) [36], the Latent Dirichlet Allocation (LDA) [12] and the *Collobert and Weston* model [17] which altogether form the foundation for most of the current approaches. Since then, several NLP tasks such as automatic caption generation [55, 56, 69], text classification based on sentiments [33] or speech recognition [48, 49], include the computation of an




(a) man - woman (b) city - zip code (c) comparative - superlative

Figure 11: Set of visualizations that show interesting patterns relying to the vectors differences between related words in an embedding space learned with GloVe. Image from [30]

embedding space for words as their first step to implement more complex behaviors.
Inspired by these techniques, our first approach was to consider the equivalence that could exist between a musical scores and textual sentences where each event (note, chord, silence) could be equivalent to a word with temporal and contextual relations. Hence, in this section, we will present the currently best-performing models (*Word2vec* [46, 47], and *GloVe* [52]) that provides state-of-the-art results for word embeddings. We will detail in the next chapter how we adapted and then extended these models for processing musical scores. Moreover, we will present the work of Boulanger-Lewandowski that reached the best performance on symbolic musical prediction, the task that we use at proxy to evaluate our approaches.

## 3.1 EMBEDDING SPACES

Embedding spaces can be learned through machine learning techniques. Indeed, we saw in the previous chapter that neural network transform an input in order to minimize the value of the loss function. Hence, we can manage to learn a transform that provide a mapping of each samples in a continuous N-dimensional space, that carry information on relationships between elements.
In the following, we will consider that the learning algorithm is fed with sentences composed of words such that $s = \{x_1 ... x_n\}$. In that case, a word $w_t$ is said to be in a context $c = \{w_{t-p}, ... w_{t-1}, w_{t+1}, ... w_{t+p}\}$. We will talk about the past context of $w_t$ as $\{w_1, ..., w_{t-1}\}$ and the future context as $\{w_{t+1}, ..., w_{t+n}\}$.

### 3.1.1 *Word2vec*

There are two critical aspects of learning embedding spaces that allows to produce interesting embeddings and also evaluate their quality. First, the training objective of the corresponding learning algorithms should make it effective for encoding general semantic relationships. Second, the computational complexity of such an objective should be low for this task, while providing an efficient coding scheme. Hence, the idea is to learn a model $f(w_t, ..., w_{t-n+1}) = \hat{P}(w_t \mid w_1^{t-1})$ that is decomposed in two parts. First, a mapping $C$ from any words to a real vector $C(i) \in \mathbb{R}^m$ that represent the *distributed feature vectors*. Then, the probability function over words expressed with $C$ through a function $g$ which maps an input sequence of feature vectors $\{C(w_{t-n+1}), ..., C(w_{t-1})\}$ to a conditional probability distribution over words for the next word $w_t$. The $i$-th element of the output of $g$ determines the probability $\hat{P}(w_t \mid w_1^{t-1})$ [8]. This architecture is depicted Figure 12.

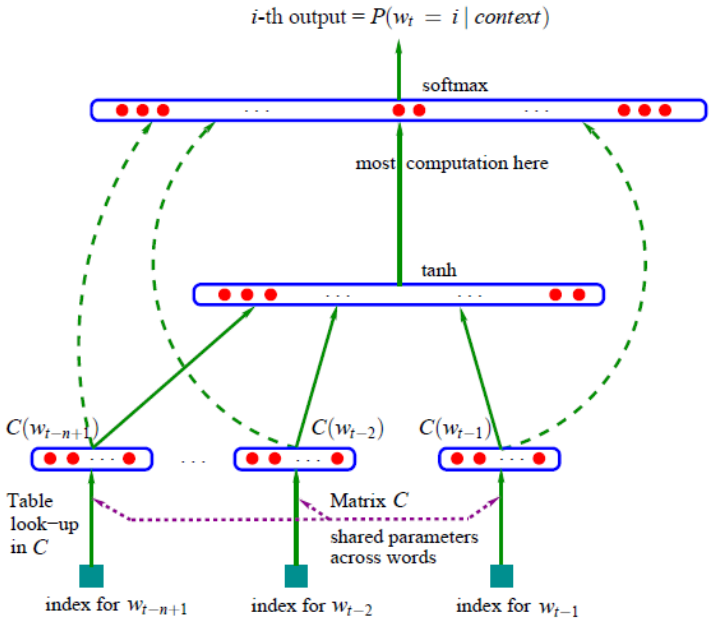



Figure 12: A classic neural architecture for word embedding. The function is defined as $f(w_t, ..., w_{t-n+1}) = g(i, C(w_{t-n+1}), ..., C(w_{t-1}))$ where $g$ is the neural network and $C(i)$ is the $i$-th word feature vector. Image from [8]

Starting from this basic neural model, *Mikolov et al.* recently proposed the Word2Vec algorithm, which is tailored around two different architectures, namely *Continuous bag-of-words* (CBOW) and *skip-gram*.

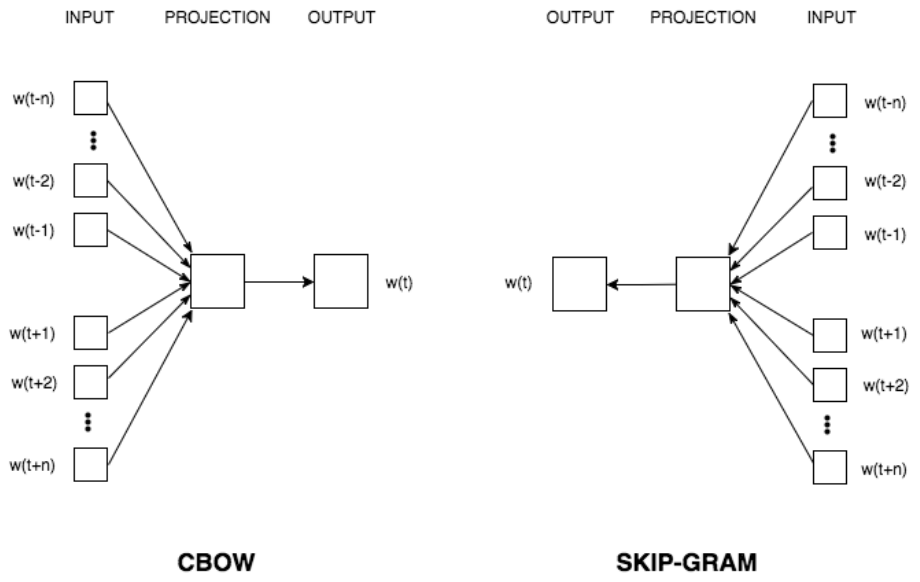



Figure 13: Continuous bag-of-words and Skip-gram architectures for Word2vec (*Mikolov et al.*, 2013). The model tent to predict a given word $w_t$ from a context composed by $n$ words before and after the target.

*Continuous bag-of-words*

The training objective of this architecture is to predict a given word $w_t$ from a context. The main idea behind the model is to use information from the past and future contexts of given word. Thus, the network take as input both the $n$ words before and after the target word $w_t$ and fine-tune its parameters $\theta$ in order to output the right prediction (see Figure 13).

Therefore the objective function that the network maximizes is defined as follows

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t \mid w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n}) \tag{12}$$

We can see that this equation simply means that we try to maximize the log-likelihood over the whole dataset of words inside their respective context, both in the past and the future.

One of the most interesting properties obtained from these concerns the distance relationships obtained in the embedding space. Hence, to show the quality of their results, the authors perform simple algebraic operations directly on the vector representation of words. For example, they compute vector $X = vector('biggest") - vector("big") + vector("small")$ and they search in the vector space for the word closest to X measured by cosine distance. If this word is the correct answer (here, the word "smallest") the operation is counted as a correct match. By doing so on several examples, they obtain a

score that reflects the efficiency of the embedding.
This model is trained on a dataset of 783 million words, in order to obtain an embedding space representation of 300 dimensions, with a context size of 10 words. Based on the *superlative task* defined on a set of 10,000 triplets, the accuracy reached by this model is 36.1%.

*Skip-gram*

While the CBOW model uses the whole context around a given word to predict it, the skip-gram model task the opposite task. Hence, the model relies on a single given word input, and tries to predict its whole context words, both in the past and future (see figure 13).
Therefore, the objective of the skip-gram is to maximize the probability of a complete context (represented by the $n$ surrounding words to the left and to the right), given that we observe a particular target word $w_t$. Consequently, the objective to maximize is given by the log-likelihood over the entire dataset of $T$ words

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\leqslant j\leqslant n,\neq 0} \log p(w_{t+j} \mid w_t) \tag{13}$$

In order to compute a given context word probability $p(w_{t+j} \mid w_t)$, the skip-gram architecture use the softmax function. In the previous (CBOW) model, this is defined for a word given its past context

$$p(w_t \mid w_{t-1}, \cdots, w_{t-n+1}) = \frac{exp(h^\top v'_{w_t})}{\sum_{w_i \in V} exp(h^\top v'_{w_i})} \tag{14}$$

Where $v'_{w_t}$ is the output embedding of word $w$ and $h$ is the output vector of the penultimate layer in the neural language network. In the skip-gram model, instead of calculating the probability of the target word $w_t$ given its previous words, the model computes the probability of a context word $w_{t+j}$ given $w_t$. Moreover, as the skip-gram model does not use an intermediate layer, in this case, $h$ simply becomes the word embedding $v_{w_t}$ of the input word $w_t$, which lead to the following equation

$$p(w_{t+j} \mid w_t) = \frac{exp(v_{w_t}^\top v'_{w_{t+j}})}{\sum_{w_i \in V} exp(v_{w_t}^\top v'_{w_i})} \tag{15}$$

Even though the task might seem extremely hard to learn, this model was shown to strongly outperform CBOW. Indeed, on the same dataset and same parameters, the accuracy for the superlative task reaches 53.3%.

### 3.1.2 *GloVe*

The main idea behind Global Vectors for word representation (GloVe) is that the ratio between the co-occurrence probabilities of a word given two

| Probability and Ratio | $k = solid$ | $k = gas$ | $k = water$ | $k = fashion$ |
|---|---|---|---|---|
| $P(k\|ice)$ | $1.9 \times 10^{-4}$ | $6.6 \times 10^{-5}$ | $3.0 \times 10^{-3}$ | $1.7 \times 10^{-5}$ |
| $P(k\|steam)$ | $2.2 \times 10^{-5}$ | $7.8 \times 10^{-4}$ | $2.2 \times 10^{-3}$ | $1.8 \times 10^{-5}$ |
| $P(k\|ice)/P(k\|steam)$ | 8.9 | $8.5 \times 10^{-2}$ | 1.36 | 0.96 |

Table 1: Co-occurrence probabilities for target words ice and steam with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam. (Table and text from *GloVe: Global Vectors for Word Representation* [52])

other words might contain significantly more information than the single co-occurrence probabilities separately (this concept is better detailed in Table 1). Hence, it is this ratio that the network will try to encode as a vector representation. Therefore, the input is no longer a stream of words defined by a sliding window but rather a complete word-context matrix of co-occurrences.
To train the model, the authors propose a weighted least square objective $J$ that directly aims to minimize the difference between the dot product of the embedding representation of two words and the logarithm of their number of co-occurrences

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^\top \hat{w}_j + b_i + \hat{b}_j - \log X_{ij})^2 \tag{16}$$

where $w_i$ and $\hat{w}_i$ are the embedding vectors of word $i$ and $j$ respectively, $b_i$ and $\hat{b}_j$ are the biases of word $i$ and $j$ respectively, $X_{ij}$ is the number of times word $i$ occurs in the context of word $j$, and $f$ is a weighting function that allows to assign a relative importance to the co-occurrences given their frequency.
An exact quantitative comparison of GloVe and word2vec is difficult to produce because of the existence of many parameters that have a strong effect on performance (vector length, context window size, corpus, vocabulary size, word frequency cut-off). This question stirred up heated debate in the machine learning community. Despite this, GloVe consistently outperformed Word2vec by achieving better results with even faster training.

## 3.2 MUSICAL SYMBOLIC PREDICTION

In order to evaluate the quality of an embedding, we have to find a task that can reflect (even partially) the efficiency of these representation spaces. In our context, we decide to test our models on musical symbolic prediction because we can easily compare our results with the previous works that pro-

vide the best published results. Indeed, we have access to the same datasets and the accuracy measure is precisely defined. Moreover, the predictive approach also showed interesting results in the field of word embeddings [8]
In this section, we will present the current state-of-the-art methods in symbolic music prediction [14] (even though they do not rely on embeddings), and explain the evaluation methods in details.

### 3.2.1 *Different models*

Recent works [14] investigated the problem of modeling symbolic sequences of polyphonic music represented by the timing, pitch and instrumentation contained in a MIDI file but ignoring dynamics and other score annotations. In this context, they exploited the ability of the well-known Restricted Boltzmann Machines (RBM) [57] to represent a complicated distribution for each time step, with parameters that depend on the previous ones [62, 64]. To that aim, Boulanger-Lewandoswki et al. [14] propose three models and evaluate them on musical symbolic prediction task, the RBM, the Recurrent temporal RBM (RTRBM) and its generalization, the RNN-RBM.

RBMs are special kind of neural network that can learn a probability distribution over the whole set of inputs. There are composed of two layers of units referred to as the "visible" and "hidden" units $v$ and $h$ that have symmetric connection between them and no connection between units within a layer. The joint probability of a given $v$ depending of the inputs and $h$ is

$$P(v, h) = \frac{exp(-b_v^T v - b_h^T h - h^T W v)}{Z} \tag{17}$$

where $b_v$, $b_h$ and W are the model parameters and Z is a normalizing function that ensure the probability distribution sums to 1.
Inference in RBMs consists of sampling the $h_i$ given $v$ (or the $v_j$ given $h$) according to their conditional Bernoulli distribution defined as

$$\begin{aligned} P(h_i = 1 \mid v) &= \sigma(b_h + Wv)_i \\ P(v_j = 1 \mid h) &= \sigma(b_v + W^T h)_j \end{aligned} \tag{18}$$

where $\sigma(x) \equiv (1 + e^{-x})^{-1}$ is the sigmoid function.

The Recurrent Temporal Restricted Boltzmann Machines, developed by Sutskever et al. in 2008 [61], is a sequence of RBMs with one at each time step. Hence, its parameters become time-dependent and denoted $b_v^{(t)}$, $b_h^{(t)}$, $W^{(t)}$. There depend on the sequence history at time t. Finally the RNN-RBM is a generalization of the RTRBM.

### 3.2.2 *Evaluation method*

To compare the models introduced previously, the authors rely on a prediction task that consists of predicting a given MIDI frame knowing its past context. Throughout the literature, this task is evaluated with four reference datasets of varying complexity. All datasets are MIDI collections split between train, test and validations sets. In order to perform an objective evaluation, we used exactly the evaluation method, sets and splits as described by [14].

PIANO-MIDI.DE is a classical piano MIDI archive introduced by Poliner & Ellis [54].

NOTTINGHAM is a collection of 1200 folk tunes[1] with chords instantiated from the ABC format.

MUSEDATA is an electronic library of orchestral and piano classical music from CCARH[2].

JSB CHORALES is the entire 382 four-part harmonized chorales by J.S Bach introduced by Allan & Williams [1]

In order to evaluate the success of various models, the *frame-level accuracy* [6] of the prediction is computed for each MIDI frame in the test set. This measure was specifically designed for evaluating the prediction of a sparse binary vector. Indeed, a purely binary measure corresponding to either a perfect match between the prediction and the groundtruth, or an error in any other case, might not reflect the true success of the underlying algorithm. For that reason, the frame-level accuracy measure is usually used to alleviate the problem.
To obtain this measure, we first compute three variables depending on the prediction vector and the groundtruth one. The *true positives* TP is the number of 1 (active notes) that the model predicts correctly, the *false positives* FP is the number of 1 which should be 0, and the *false negatives* FN is the opposite (0 that should be 1). Note that the authors do not take into account the *true negatives* value due to the sparsity of a pitch class vector. Indeed, this property of sparsity leads to a very high number of true negatives (a wide percentage of 0 in both vectors), which might skew the measure by artificially inflating the success of these algorithms. Hence, from these three variables, we can calculate an overall accuracy score for a given predicted vector as follows

$$\mathrm{Acc} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{19}$$

However, we can see that this measure fails to account for rightfully predicted rests (vectors filled with only 0), as even with a perfect prediction, we

1 ifdo.ca/~seymour/nottingham/nottingham.html
2 www.musedata.org

obtain $TP = 0$. To account for this case, we introduce in this work a new measure of accuracy where a term is defined specifically for the rests and the global accuracy score for $N$ predicted vectors with at least one active unit and $M$ vectors of rest becomes

$$Accuracy = \frac{1}{M+N}\left(\sum_{n=1}^{N}\frac{TP_n}{TP_n+FP_n+FN_n}+\sum_{m=1}^{M}\frac{1}{1+FP_m}\right) \tag{20}$$

This proposed measure of overall performance is now bounded between 0 and 1 where 1 corresponds to perfect prediction. We will rely on this measure to evaluate the performance of different models in the subsequent chapter. The results reported by Boulanger-Lewandoswki are presented in Table 3.

# 4

# SYMBOLIC MUSICAL EMBEDDING SPACE

We now introduce our approaches to obtain a symbolic musical embedding space automatically through learning algorithms. As discussed in Chapter 2, the results of learning algorithms highly depend on the dataset and the tasks that we use to train them. Hence, in the first section, we will present all the datasets we used along our work. Then, we introduce our main contribution, by proposing two main approaches. The first one is based on previous works in the NLP field that yielded very efficient word embeddings, as introduced previously (see Section 3.1). Starting from the analogy that musical scores could be considered as textual sentences with temporal and contextual relationships alike, we tried to adapt the major word embedding models to musical data.
Nevertheless, some critical differences still exist between music and text. Indeed, a text is composed of a very large corpus of words with very rare occurrences (except certain words like "the" or "a"). Oppositely, a musical score is composed of a little number of elements (notes and chords) that occur relatively often. Moreover, even though the contextual content of a text could be akin to its temporal component, there is a crucial temporal aspect in music that is defined by its rhythm (adding the duration dimension to an otherwise set of ordered elements). Indeed, the perception of a musical phrase highly depends on the duration of each event and not only on their sequential position. For these reasons, we developed a new model based on a CNN (see Section 2.2.2) in order to capture relative harmonic relationships independently of the global pitch of the notes. Furthermore, we augment this approach with a LSTM (see Section 2.2.3) to account for more complex temporal dependencies.

## 4.1 MATERIALS AND METHOD

### 4.1.1 *Datasets*

In order to train any neural network, we need a large amount of data that represents the information we want to learn (see Chapter 2). Therefore, in our context, we would require as much machine-readable musical scores as

possible. Hence, the first step of our research was to collect as much relevant MIDI files as possible. Overall, 93,237 tracks have been gathered, MIDI files taken from five different large databases. We collected 3,187 files from GreatArchive, 46,918 files from IMSLP, 14,415 files from Kunstderfuge, 675 files from Mutopia and finally 28,042 files from in-house IRCAM collections. Altogether, these tracks represent a total of 15,184 different composers from various eras. Unfortunately, the annotation and naming schemes for composers and track names vary widely amongst the different databases, which made the merging process difficult. To alleviate this issue, we first computed the Levenshtein distance between all composer names and used a .2 threshold of differences, under which the composers were considered equivalent. These automatic assignments were then manually checked in order to correct or remove any inaccuracy. Finally, every MIDI channels were filtered using the same name-matching procedure in order to ensure that they were linked correctly to a pre-defined set of instruments.
We note here that the word embedding models presented earlier (see Chapter 3), which we will adapt to musical data are trained with several billions of token, which is several orders of magnitude above our current data collection. Furthermore, collecting a large set of clean MIDI scores is more difficult than English text which is plentiful on large websites such as Wikipedia. These issues renders impossible the possibility to attain a dataset as wide as those used in NLP research. Hence, to fill this lack of information we used the so-called *data augmentation* method. This technique consists in applying a set of mathematical transformations that preserve structural properties of the input data. Therefore, we also apply transpositions to the dataset, ranging from six half tones higher to six lower. That way, we multiply the cardinality of our dataset by twelve (which still remains largely under the cardinality of NLP datasets).

The MIDI format encodes all the information about the scores that we need but it is not ideally suited to be manipulated inside learning algorithms. Hence, we imported all the MIDI files in a *piano-roll* representation matrix for each channel. This means that, for each instrument and at every time step (given a certain time quantification), we encode a 128-dimensional vector where each value corresponds to the dynamic of a note ranging from C-2 to G8. Therefore, each dimension in this vector takes a value between 0 and 127.

## 4.2 WORD EMBEDDING ALGORITHMS ADAPTED TO MUSICAL SCORES

The two models introduced previously (see Section 3.1) perform very well when applied to learning word embedding spaces. The basic assumption of *Word2vec* [46, 47] is that the semantic meaning of a word depends on the context where it occurs. Despite a different approach based on the co-

occurrences of each word, *GloVe* [52] depends on the same overall assumptions. Therefore, our first objective was to evaluate this context-dependent hypothesis to the application of musical scores. Hence, we adapted our data in order to use it with these algorithms, while performing some adjustments on the algorithms themselves. In this section, we introduce our methodology and then the results obtained. Note that we separate these results in two different parts, one specifically for the embedding and the other including the prediction.

### 4.2.1 *Methodology*

As we will be relying on existing models, we first need to process the input data to obtain the same representation used in the original works. *Word2vec* relies on a text file (where each line represent a separate sentence) and *GloVe* uses the co-occurrence matrix (between all the words in the corpus) as input. In order to provide a text input based on our MIDI scores, we decided to consider each chord (pitch vectors) into a specific word. For instance, a vector with non-zero values at the 37th, 41st and 44th positions (counting from 1 to 128) becomes the string of characters "C1E1G1". Hence, for each new event, we produce a word representation for the corresponding chord and then separate each word with a space, leading to a text-like representation which transcribes the musical scores as sets of sentences. However, due to the time quantification, MIDI events that last more than one time steps lead to successions of identical words. This contradicts with the fixed-context hypothesis for textual analysis used by the algorithms that we try to adapt. Indeed, numerous repetitions would lead to useless contexts that could hamper the learning phase and skew the results. To alleviate this issue, we keep only one occurrence per musical event in order to obtain our text-based musical score. It is important to note that, in doing so, we discard every information regarding rhythm and base our musical embedding solely on the sequential position of each chord. We rely on the same simplification in order to construct the global co-occurrence matrix for GloVe.

### 4.2.2 *Embedding*

Once the training phase completed, we obtain a vector representing the position in the embedding space for each musical events encountered in the training corpus. In order to evaluate if the model has successfully learn some musical concepts, we firstly project the embedding through the t-SNE algorithm to search interesting patterns that could reflect semantic relationships. The projection of a large part of the objects can highlight the global structure of the embedding when the representation of only few given symbols can show some more specific features. We present this analysis for our

embedding space obtained from the Skip-gram architecture of Word2vec in Figure 14.

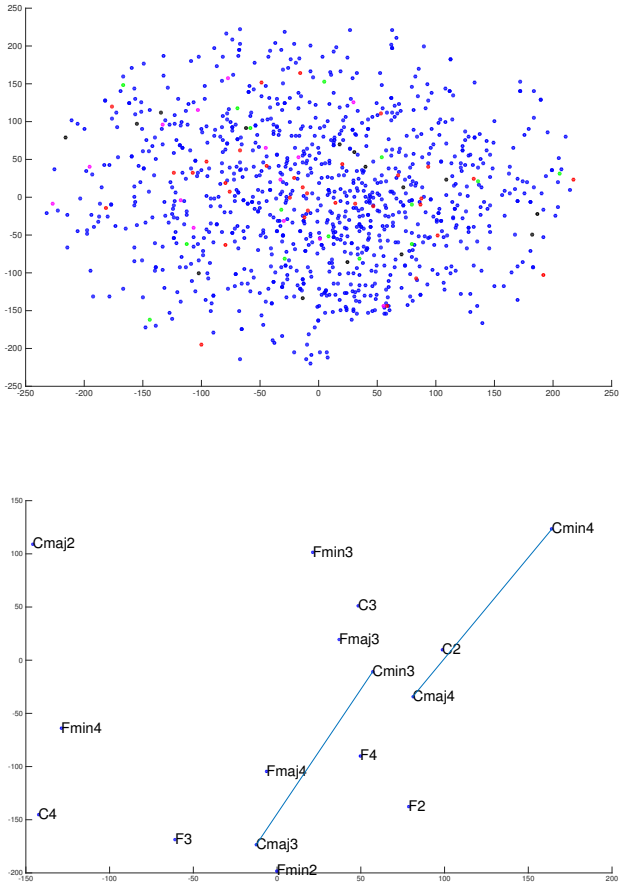



Figure 14: Embedding space learned with Word2vec (with the skip-gram architecture) and projected through the t-SNE algorithm. On the top, we can see the global structure of our embedding. The red points represent the note alone, the magenta points represent the chords containing a minor third, the black points represent the chords containing a major third and the green points represent the chords containing a fifth. On the bottom, we can see a representation of few particular points that could exhibit geometrical relations. Even if this embedding does not provide a representation as clean as we expected, we can notice that the distances between the C4 major chord and the C4 minor chord and between the C3 major chord and the C3 minor chord are equal.

We can see that we do not have the clean relations shown in text applications. Indeed, we expected the emergence of different clusters in the global representation but the points are organized sparsely. However, in the restricted projection, we can see that the distance between the C4 major and minor chords is the same than the one between the C3 major and minor

chords. this pattern might show a beginning of learning on musical concepts, as during the training we did not provide any supervised information about what major and minor mean.

### 4.2.3 *Prediction*

In order to properly compare the different models, we rely on the same tasks, datasets, splits (train, test, valid see Section 2.1.2) and accuracy measure defined in the symbolic music prediction literature [14]. Hence, we use the prediction task and the four datasets presented in Section 3.2.2. We implemented these tasks and evaluated both of the state-of-the-art models at different context window sizes. We report the results of our experiments on the JSB Chorales dataset in table 2. Through this results, we can see that GloVe did not perform well on the prediction task. Its score being close to the random function, we can infer that the ability of this model to learn semantic relationships is close to zero. Several reasons may be responsible for this failure. First, this model was tailored for a very large amount of tokens and, despite our data augmentation process, we used one hundred times less of samples than in the original shape. Moreover, besides the lost of temporal informations, some problem may have occurred during the co-occurrence computation.
On the other hand, the prediction score of Word2vec are rather bad but two times better than the random function, which might show a beginning of learning as the embedding leaded us to believe.

| | Random | Glove | Word2vec |
|---|---|---|---|
| Context size | ACC % | ACC % | ACC % |
| 5 | 4.42 | 4.89 | 6.94 |
| 7 | 4.42 | 4.14 | UC |
| 9 | 4.42 | 4.41 | UC |

Table 2: Expected accuracy for both word embedding models adapted to musical scores in the symbolic prediction task on the JSB Chorales dataset depending on the context windows size. The results noted UC are still under calculation.

## 4.3 NEW MODEL - CNN-LSTM

Besides the efficiency of recently developed word embedding algorithms, we have seen in our first experiment that the possible analogy between musical scores and textual sentences might be insufficient to correctly learn on musical objects. Oppositely, we introduce in this section a new model, that takes

its roots from the critical differences that exist between textual and musical data. Indeed, a text is composed by a very wide variety of symbols (words) that appear sparsely across the data, while musical scores are defined by few symbols that re-occur frequently along the score. Moreover, the crucial notion of octave in music does not exist in text, while we would expect any musical embedding space to deal with it adequately. Another musical aspect that the word embeddings algorithms are not able to handle correctly is the rhythmic information. In these models, there are no components that aim to capture information about time dependencies. Given these observations, we tried to build a model that could fill all these lacks through two main modules, a CNN (for handling transposition-invariance) and an LSTM (for targeting temporal relationships). (Section 2.2).

### 4.3.1 *Architecture*

We have seen previously that a CNN is able to extract recurrent patterns in a given matrix through the convolution of small kernels across this matrix. The idea here is to use this property in order to learn different chord patterns in an octave-invariant manner. In other words, we expect that a given transformation layer will activate the same features from the input vectors encoding "C1F1A1" as for the one encoding "C2F2A2". To do so, we rely on a musically-motivated convolutional architecture with the kernel sizes set to 12 in height (pitch dimension) and 1 in width (time dimension), while having a step size in the pitch dimension set to 1. Finally, a zero-padding is applied to this dimension in order to account for lowest and highest notes. Through this architecture, we influence the model to learn recurrent features that would be octave-invariant.

Here, we propose two different architectures for the convolutional networks. First, we build the simplest CNN possible by relying on a single convolutional layer, followed by a *Batch-Normalization* (BN) and a non-linear transform. We introduce succinctly here the *batch normalization* principle [29]. Most machine learning models rely on batches of data instead of single inputs in their learning phase (to improve computational efficiency and gradient stability). However, each batch might present different statistics (correlated examples), which could skew the learning. By shifting these inputs to zero-mean and unit variance transforms with respect to a given batch, this problem is largely avoided. Furthermore, this improved method allows to use a higher learning rate and potentially provide a faster learning.
The second architecture is a more complicated model called *DenseNet* that have been developed very recently by Huang et al. [27]. Starting from the observation shown by recent works, that convolutional networks can be more efficient to train if they contain shorter connections between layers (called *residual connections*), the authors introduce a network which connects

each layer to every other following layers in a feed-forward fashion. Hence, for a convolutional network with K layers, the DenseNet has $K(K+1)/2$ direct connections instead of K. Besides significant improvements over the state-of-the-art object recognition benchmark tasks, these networks *"alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters"*[1]. This architecture is depicted in Figure 15.

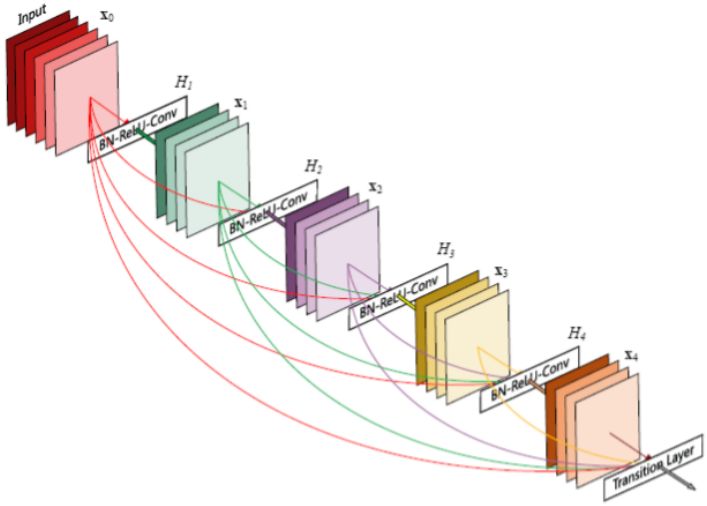



Figure 15: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input. Image and caption from [27]

This architecture can be adapted depending on a wide set of hyper-parameters, listed in [27]. These parameters include the number of dense blocks (successive fully-connected layers) D, the number of filters in the first block $k_0$, the growth rate k which regulates the number of kernels in the $l^{th}$ layer to $k \times (l-1) + k_0$ and finally the reduction factor r which reduces the number of filters in transition layers. For our experiments, we set $D = 1$, $k_0 = 200$, $k = 12$ and $r = 0.5$.

The points in our embedding space will simply be defined as the activation outputs of this Dense CNN, by producing a different N-dimensional vector for each input chord. However, the Dense CNN outputs very large activation matrices (feature maps), whereas we would like to obtain a space with a reasonable dimensionality. Therefore, the Dense CNN is followed by a fully-connected MLP (see Section 2.2.1) which allows to constrain the target dimensionality of the embedding. Here, we use a MLP with a depth of three layers and a number of units of 1500, 500, and 50 respectively.
As discussed previously, we also want to capture the information about the

1 Huang et al. in [27]

time dependencies between different input frames. To that aim, we add an LSTM network (Section 2.2.3) after the MLP to target specifically these relationships. Hence, the LSTM uses as input the compact embedding representation of each input frame outputted by the first part of the network. This kind of representation might lead to an easier learning than with full MIDI vectors, as the corresponding space has a lower dimensionality but most importantly should exhibit relationships between musical objects. In our case, we propose an LSTM network with two layers and 1500 hidden units for each layer. The complete architecture of our model is depicted Figure 16.

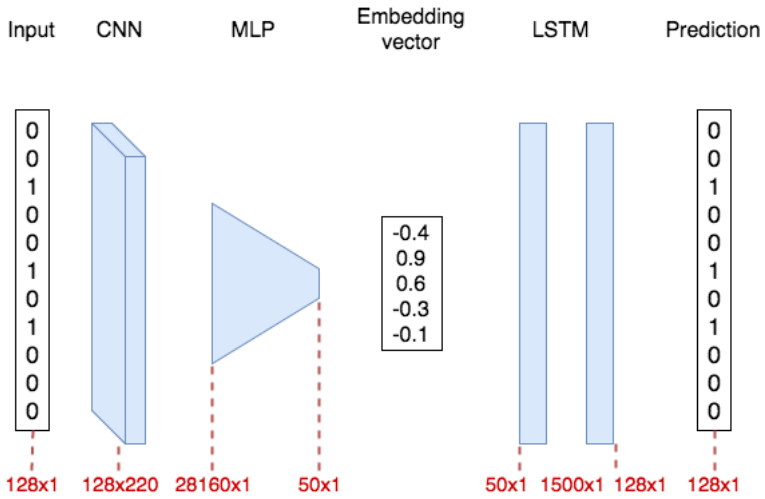



Figure 16: CNN-LSTM architecture for learning a 50-dimensional musical embedding space. The red numbers show the sizes of the data at each step of the feed-forward pass.

To train our model, we propose two methods. The first one consists in training all the modules of the model together and therefore optimize the whole parameters during the same pass in order to minimize the loss of the final prediction. However, this model being quite complex (composed by a large number of parameters), it can make the training phase very long and unstable. Hence, we propose a second approach which is to pre-train the CNN before doing the complete training. To do so, we start by training the CNN in an encoder/decoder manner (similar to the embedding task). This means that we force the Dense CNN to encode the input frames as a vector of the desired dimensions and then try to retrieve the original vector by applying the inverse transform. Then, the loss is defined by the distance between the inputs and the reconstructed vector. Thus, by minimizing this loss, the CNN will tend to project the data in a lower dimensional space that carry enough information to be able to retrieve the original representation. Once this CNN is trained (when an arbitrary threshold on the loss value is reached), we train the entire network on the previously defined prediction task.

### 4.3.2 *Embedding*

With this architecture, the CNN (inside the whole network) can be seen as an embedding that will be adapted based on temporal relationships through the LSTM layers. The projection of the embedding show similar results than the one learned through adapted word embedding models (see Figure 14). We intend to improve our visualization method in order to highlight more interesting patterns in a future work.

### 4.3.3 *Prediction*

We implemented our model and evaluated it on the same task and datasets as previously introduced. We report here the global results of expected accuracy, while comparing to the current state-of-the-art in Table 3.

| Model | Piano-midi.de | Nottingham | MuseData | JSB Chorales |
|---|---|---|---|---|
| | ACC % | ACC % | ACC % | ACC % |
| Random | 3.35 | 4.53 | 3.74 | 4.42 |
| 1-Gram (Gaussian) | 6.04 | 21.31 | 7.87 | 17.41 |
| GMM + HMM | 7.91 | 59.27 | 13.93 | 19.24 |
| RBM | 5.63 | 5.81 | 8.19 | 4.47 |
| GloVe-Music | UC | UC | UC | 4.89 |
| Word2vec-Music | UC | UC | UC | 6.94 |
| CNN-LSTM (Pre-train) | UC | UC | UC | 13.94 |
| CNN-LSTM | UC | UC | UC | 22.76 |
| RTRBM | 22.99 | 75.01 | 30.85 | 30.17 |
| RNN-RBM | **28.92** | **75.40** | **34.02** | **33.12** |

Table 3: Expected accuracy for various musical models in the symbolic prediction task on four different datasets [14]. The results noted UC are still under calculation.

This results show that our proposed models perform a lot better than GloVe and Word2vec on the prediction task. However, we do not reach the higher scores of the state-of-the-art. Moreover, our pre-training method do not improve the prediction results yet but a lot of works still remain to be done in order to improve our encoder/decoder training phase such as fine-tuning hyper-parameters or choosing the dataset and the loss function. However, in comparison with the RBM-based models that are very efficient on the prediction task, our model is very simple in terms of number of parameters, and also provide an embedding in the middle of the system.

# 5

# APPLICATIONS

Many applications could be developed based on an adequate embedding space for symbolic music. Here, we introduce different proposals that we implemented along this internship, that are split in two main types of applications. The first one concerns the *visual* applications that we can obtain by applying the t-Stochastic Neighbors Embedding (t-SNE) algorithm [41]. Despite the fact that our embeddings are 50-dimensional vectors, this algorithm allows to visualize the learned spaces in a two or three-dimensional map keeping the main distance relationships between points in the original space. That way, we can analyze the potential correlations between musical symbols that emerge from the spaces, and obtain an efficient analysis tool. Secondly, we introduce different propositions to produce musical applications such as scores transformation or even direct generation.

## 5.1 VISUAL

### 5.1.1 *t-SNE visualization of symbols*

A possible approach to analyze musical concepts through an embedding space is to visualize it and to evaluate the different geometric relationships that could emerge between musical objects, and try to relate them to higher-level abstractions. However, our embedding spaces being high-dimensional, we have to rely on the t-SNE algorithm that produces a projection of such spaces into two or three-dimensional maps while retaining the global structure of the original space. An example of t-SNE visualization is provided in Figure14

In order to go further with these types of musical analysis, we developed an *interactive embedding* system, allowing to dynamically exhibit different types of relationships. Given some group of symbols (for instance notes alone, chords containing a third or a fifth) this tool provides a quick and clear view of the positions of each object with regards to its group membership. Moreover, the system can outline specific harmonic relationships that will be put forward when we parse across the different symbols. Besides

analysis, this prototype interface could be used in a pedagogical manner, showing in a very visual way some musical pieces or concepts.

### 5.1.2 *Track paths*

In order to analyze a particular musical piece, we can project every symbols that composed it and produce a sequential path that link them in the embedding space. In doing so, we obtain the complete *temporal trace* taken by the given track in the embedding. By analyzing the geometric properties of this path, we can infer some knowledge about the composition process or even discover temporal and harmonic structures of this musical piece.

## 5.2 MUSICAL

### 5.2.1 *Symbolic generation / transformation*

In this section, we investigate the use of these spaces as a musical generation tool. Hence, we developed a program that can modify any given track by processing a geometrical transformation of its path inside the embedding space, rather than working on the raw data itself. The intuition behind this system is that we could obtain different musical pieces following the same higher-level structure by applying sets of linear operations (such as translation or rotation), on every symbols that composed the piece. Then, we replace all the original MIDI events with the *most similar* (nearest neighbor) of the transformed points as captured by the cosine distance in the embedding space.

# 6

# CONCLUSION

## 6.1 DISCUSSION

The goal of this internship was to develop new representations of symbolic music in an empirical manner through a machine learning framework.

Firstly, we focused on the adaptation of two efficient words embedding models to musical symbols. We tested the performance of the resulting musical embeddings through two different approaches. First, we started by using the t-SNE algorithm in order to project our high-dimensional embedding space into a 2-dimensional map and, then, searching possible geometric relationships between objects that could exhibit a learning of musical semantic concepts. On the other hand, we evaluated the embeddings through a prediction task which, once again, reflects the ability of the space to represent musical data in a meaningful manner. For both of these tasks, the musical embeddings that we directly adapted from the NLP field show limited to poor results and also, did not appear to be efficient in our context. This can be possibly explained by the fact that the first hypotheses made to shape the word embedding models on textual symbols do not translate directly to musical symbols. Indeed, there are some critical features in the musical symbols that we do not retrieve in a text such as the notion of pitch class (octave-invariance) and the important temporal dependencies defined by the rhythm.

Starting out from these observations, we proposed a new model able to fill these gaps. To that end, we used a Convolutional Neural network (CNN) followed by a Long Short Term Memory (LSTM) network that we trained on the same task. With the same evaluation method, we already showed the widely improved results for these types of models on the prediction task. However, it is a complex network that contain a large amount of parameters that made this training very sensitive. Hence, we proposed an alternative for the training phase which consists of alleviating the tuning of these parameters altogether, by rather first pre-training the CNN in an encoder/decoder framework. Unfortunately, this method did not improve the prediction re-

sults yet, but the pre-training phase can still largely be improved.

In order to train the most efficient models as possible, we tried to build a dataset with a large amount of MIDI files. But collecting proper musical scores is not easy, and besides several cleaning and checking processes, we believe that some errors could remain in a large part of the scores, which could skew this learning phase.

In conclusion, we succeed to build a very promising approach that aims to represent symbolic music objects in a space that carry semantic relationships between the elements. With many possible improvements, our model already shown very decent results on the hard task of predicting musical events. Moreover, we proposed several applications based on such representations that could allow both to infer knowledge on musical concepts and to increase musical creativity.

## 6.2 FUTURE WORK

After improvements on the proposed approach, we intend to combine the symbolic embedding space with audio representations. By learning joint multimodal embedding spaces, we could link together symbolic, acoustic and perceptual sources of information to disentangle the correlations that emerge from orchestral music. To that end, a possible method could be to use zero-shot learning [51] that is based on the idea that multiple modalities of a same object lead to widely different raw data, but still carry a common semantic content. It has been shown that these spaces provide astonishing regularities and metric relationships over semantic concepts [45] allowing for analogies, knowledge inference and missing modality generation [60] that can be exploited in our context to attain multiple musical and pedagogical goals. Furthermore, automatic inference through embedding spaces can decipher the signal-symbol relationships to provide optimal features for orchestration, by targeting correlations existing in the work of well-known composers.

# BIBLIOGRAPHY


[1] Moray Allan and Christopher Williams. "Harmonising chorales by probabilistic inference." In: *Advances in neural information processing systems*. 2005, pp. 25–32.

[2] Aurélien Antoine and Eduardo R Miranda. "Towards intelligent orchestration systems." In: *11th International Symposium on Computer Music Multidisciplinary Research, Plymouth, UK*. 2015, p. 124.

[3] Willi Apel. *The notation of polyphonic music, 900-1600*. 38. Medieval Academy of Amer, 1961.

[4] Gérard Assayag, Camilo Rueda, Mikael Laurson, Carlos Agon, and Olivier Delerue. "Computer-assisted composition at IRCAM: From PatchWork to OpenMusic." In: *Computer Music Journal* 23.3 (1999), pp. 59–72.

[5] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. "Soundnet: Learning sound representations from unlabeled video." In: *Advances in Neural Information Processing Systems*. 2016, pp. 892–900.

[6] Mert Bay, Andreas F Ehmann, and J Stephen Downie. "Evaluation of Multiple-Fo Estimation and Tracking Systems." In: *ISMIR*. 2009, pp. 315–320.

[7] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. "A neural probabilistic language model." In: *Journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[9] Yoshua Bengio et al. "Learning deep architectures for AI." In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[10] Louis Bigo and Moreno Andreatta. "Towards Structural (Popular) Music Information Research." In: *European Music Analysis Conference (EuroMAC 2017)*. 2017.

[11] Louis Bigo, Jean-Louis Giavitto, and Antoine Spicher. "Building Topological Spaces for Musical Objects." In: *MCM*. Springer. 2011, pp. 13–28.

[12] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation." In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[13] Jean-Yves Bosseur. *Du son au signe: histoire de la notation musicale*. Editions Alternatives, 2005.

[14] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription." In: *arXiv preprint arXiv:1206.6392* (2012).

[15] Augustin Cauchy. "Méthode générale pour la résolution des systemes d'équations simultanées." In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[16] Christopher Colah. *Colah's blog*. http://colah.github.io/. Accessed: 2017-04.

[17] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning." In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 160–167.

[18] George E Dahl, Tara N Sainath, and Geoffrey 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] Jeffrey L Elman. "Finding structure in time." In: *Cognitive science* 14.2 (1990), pp. 179–211.

[20] Leonhard Euler. *Tentamen novae theoriae musicae ex certissimis harmoniae principiis dilucide expositae*. ex typographia Academiae scientiarum, 1739.

[21] Leonhard Euler. "De harmoniae veris principiis per speculum musicum repraesentatis." In: *Opera Omnia* 3.1 (1774), pp. 568–586.

[22] Michael Good et al. "MusicXML: An internet-friendly format for sheet music." In: *XML Conference and Expo*. 2001, pp. 03–04.

[23] Richard Gross. *Psychology: The science of mind and behaviour 7th edition*. Hodder Education, 2015.

[24] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[25] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[26] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001.

[27] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. "Densely connected convolutional networks." In: *arXiv preprint arXiv:1608.06993* (2016).

[28] David H Hubel and Torsten N Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." In: *The Journal of physiology* 160.1 (1962), pp. 106–154.

[29] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[30] Christopher D.Manning Jeffrey Pennington Richard Socher. *GloVe: Global Vectors for Word Representation*. https://nlp.stanford.edu/projects/glove/. Accessed: 2017-08.

[31] Andrej Karpathy and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3128–3137.

[32] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. "Multimodal Neural Language Models." In: *Icml*. Vol. 14. 2014, pp. 595–603.

[33] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Skip-thought vectors." In: *Advances in neural information processing systems*. 2015, pp. 3294–3302.

[34] Henry James Klumpenhouwer. *A generalized model of voice-leading for atonal music*. Harvard University, 1991.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[36] Thomas K Landauer. *Latent semantic analysis*. Wiley Online Library, 2006.

[37] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. "Unsupervised feature learning for audio classification using convolutional deep belief networks." In: *Advances in neural information processing systems*. 2009, pp. 1096–1104.

[38] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function." In: *Neural networks* 6.6 (1993), pp. 861–867.

[39] David Lewin. "Klumpenhouwer networks and some isographies that involve them." In: *Music Theory Spectrum* 12.1 (1990), pp. 83–120.

[40] David Lewin. "A Tutorial on Klumpenhouwer Networks, Using the Chorale in Schoenberg's Opus 11, No. 2." In: *Journal of Music Theory* 38.1 (1994), pp. 79–101.

[41] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[42] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[43] Marin Mersenne. *Harmonicorum libri xii....* 1972.

[44] Marcel Mesnage. "Entités formelles pour l'analyse musicale." In: *GENEVOIS et ORLAREY* (1997), pp. 93–109.

[45] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations." In: *Hlt-naacl*. Vol. 13. 2013, pp. 746–751.

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." In: *arXiv preprint arXiv:1301.3781* (2013).

[48] Youssef Mroueh, Etienne Marcheret, and Vaibhava Goel. "Deep multimodal learning for audio-visual speech recognition." In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 2130–2134.

[49] Kuniaki Noda, Yuki Yamaguchi, Kazuhiro Nakadai, Hiroshi G Okuno, and Tetsuya Ogata. "Audio-visual speech recognition using deep learning." In: *Applied Intelligence* 42.4 (2015), pp. 722–737.

[50] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval." In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24.4 (2016), pp. 694–707.

[51] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. "Zero-shot learning with semantic output codes." In: *Advances in neural information processing systems*. 2009, pp. 1410–1418.

[52] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global Vectors for Word Representation." In: *EMNLP*. Vol. 14. 2014, pp. 1532–1543.

[53] George Perle. *Twelve-tone tonality*. Univ of California Press, 1996.

[54] Graham E Poliner and Daniel PW Ellis. "A discriminative model for polyphonic piano transcription." In: *EURASIP Journal on Applied Signal Processing* 2007.1 (2007), pp. 154–154.

[55] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. "Generative adversarial text to image synthesis." In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 3. 2016.

[56] Zhou Ren, Hailin Jin, Zhe Lin, Chen Fang, and Alan Yuille. "Joint Image-Text Representation by Gaussian Visual-Semantic Embedding." In: *Proceedings of the 2016 ACM on Multimedia Conference*. ACM. 2016, pp. 207–211.

[57] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 791–798.

[58] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." In: *Advances in neural information processing systems*. 2014, pp. 568–576.

[59] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014).

[60] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. "Zero-shot learning through cross-modal transfer." In: *Advances in neural information processing systems*. 2013, pp. 935–943.

[61] Ilya Sutskever, Geoffrey E Hinton, and Graham W Taylor. "The recurrent temporal restricted boltzmann machine." In: *Advances in Neural Information Processing Systems*. 2009, pp. 1601–1608.

[62] Ilya Sutskever and Geoffrey Hinton. "Learning multilevel distributed representations for high-dimensional sequences." In: *Artificial Intelligence and Statistics*. 2007, pp. 548–555.

[63] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification." In: *ACL (1)*. 2014, pp. 1555–1565.

[64] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. "Modeling human motion using binary latent variables." In: *Advances in neural information processing systems*. 2007, pp. 1345–1352.

[65] Rainer Typke, Remco C Veltkamp, and Frans Wiering. "Searching notated polyphonic music using transportation distances." In: *Proceedings of the 12th annual ACM international conference on Multimedia*. ACM. 2004, pp. 128–135.

[66] Alexandra Uitdenbogerd and Justin Zobel. "Melodic matching techniques for large music databases." In: *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*. ACM. 1999, pp. 57–66.

[67] Esko Ukkonen, Kjell Lemström, and Veli Mäkinen. "Geometric algorithms for transposition invariant content-based music retrieval." In: (2003).

[68] Anatol Vieru. *The Musical signification of Multipplication by 7. Diatonicity and Chromaticity*. 1995.

[69] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. "Show and tell: A neural image caption generator." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3156–3164.