Dancing with Deep Learning

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica d’Enginyeria de
Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by
Gisela Arnal Romero

In partial fulfilment
of the requirements for the degree in
Audiovisual Systems ENGINEERING

Advisors: Elisa Sayrol and Josep Vidal

Barcelona, June 2016
ABSTRACT

Art and technology are in constant synergetic expansion. Willy Barleycorn - *Machine Learning in Dance*.

When art uses technology as a way of creation by the hand of the own artist it comes a synergy with great potential. “Dancing With Deep Learning” is born from the convergence between technology and art in order to create a machine-human collaboration. In such a manner, the system is a catalyst of inspiration for the dancer and choreographer, allowing them to expand their creations inside the artistic area. This final degree thesis exposes a generation system of new choreographic material in the own style of the dancer as a way to record her artistic repertory or proposing new material in front of the artistic lack. Inside of the core of “Dancing with Deep Learning” we find a Recurrent Neural Network trained with given data by the movement of the dancer that is also capable of generating new sequences of dance.
RESUM

Art and technology are in constant synergetic expansion. Willy Barleycorn - 
*Machine Learning in Dance.*

Quan l’art utilitza la tecnologia com a medi de creació de la mà dels propis artistes sorgeix una sinèrgia amb gran potencial. Dancing with deep learning neix de la convergència entre tecnologia i art amb la finalitat d’una col·laboració màquina/humà. Així doncs, el sistema és un catalitzador d’inspiració per al ballarí i coreògraf permetent expandir les seves creacions dins de l’àmbit artístic. El present treball de final de grau presenta un sistema de generació de nou material coreogràfic dins de l’estil del ballarí amb la finalitat d’ampliar el seu repertori artístic o propiciar-li nou material enfront de la manca d’aquest últim. En el nucli de Dancing with Deep Learning es troba una xarxa neuronal recurrent entrenada amb dades proporcionades pel moviment del ballarí, per generar noves seqüències de ball.
RESUMEN

Art and technology are in constant synergetic expansion. *Willy Barleycorn - Machine Learning in Dance.*

Cuando el arte utiliza la tecnología como medio de creación de la mano de los propios artistas surge una sinergia con gran potencial. Dancing with deep learning nace de la convergencia entre tecnología y arte con el fin de una colaboración máquina/humano. Así pues, el sistema es un catalizador de inspiración para el bailarín y coreógrafo permitiendo expandir sus creaciones dentro del ámbito artístico. El presente trabajo de final de grado presenta un sistema de generación de nuevo material coreográfico dentro del estilo del bailarín con el fin de ampliar su repertorio artístico o proporcionarle nuevo material frente a la carencia del mismo. En el núcleo de Dancing with Deep Learning se encuentra una red neuronal recurrente entrenada con datos proporcionados por el movimiento del bailarín, para generar nuevas secuencias de baile.
ACKNOWLEDGEMENTS

The present development of the thesis would not have been possible without the help of all the amazing friends, dancers, teachers and assistants who have helped make this challenge possible.

I am in profound thankfulness to all of them for helping me all over the process. I would like to express special gratitude to Ainhoa Kake, for assisting me for many hours as possible during the database recording, for all the water bottles that she threw me and to force me to make this thesis. I specially want to thank Albert Aparicio, who I feel in debt, for showing me the “deep” world of Tensorflow. I think I could not be happier to having had by my side during all this work my mentors Elisa Sayrol and Josep Vidal. Thank you for the given swing classes. I also have to thank Antonio Bonafonte, for being such an incredible teacher and simplify my work.

I wish to extend my profound sense of gratitude to my parents for all the sacrifices they made during my research and also for providing me with moral and economic support as well as encouragement whenever required.
REVISION HISTORY AND APPROVAL RECORD

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>05/05/2017</td>
<td>Document Creation</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>Document Revision</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Document Revision</td>
</tr>
</tbody>
</table>

Document distribution list

<table>
<thead>
<tr>
<th>Name</th>
<th>e-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gisela Arnal Romero</td>
<td><a href="mailto:gigisela.28@gmail.com">gigisela.28@gmail.com</a></td>
</tr>
<tr>
<td>Elisa Sayrol</td>
<td><a href="mailto:elisa.sayrol@upc.edu">elisa.sayrol@upc.edu</a></td>
</tr>
<tr>
<td>Josep Vidal</td>
<td><a href="mailto:josep.vidal@upc.edu">josep.vidal@upc.edu</a></td>
</tr>
</tbody>
</table>

WRITTEN BY:                  REVIEWED AND APPROVED BY:

<table>
<thead>
<tr>
<th>Date</th>
<th>Date</th>
<th>Name</th>
<th>Position</th>
<th>Name</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/05/2017</td>
<td>-</td>
<td>Gisela Arnal Romero</td>
<td>Project Author</td>
<td>-</td>
<td>Project Supervisor</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

ABSTRACT ................................................................. 2
RESUM ................................................................. 3
RESUMEN ............................................................. 4
ACKNOWLEDGEMENTS .................................................. 5
REVISION HISTORY AND APPROVAL RECORD .................. 6
LIST OF FIGURES ...................................................... 9
LIST OF TABLES ......................................................... 11
LIST OF TERMS AND ABBREVIATIONS ............................. 11

1 Introduction 1
1.1 Thesis definition and goals ................................. 1
1.2 State of the art .................................................. 2
1.3 Thesis framework ............................................. 3
1.4 Methods and procedures ................................... 3
1.5 Thesis workplan .............................................. 4
1.6 Deviations from initial plan ............................... 8
1.7 Thesis structure .............................................. 8

2 Kinect for Windows V2 9
2.1 Specifications .................................................. 9
  2.1.1 Hardware .................................................... 9
  2.1.2 Software - Visual Studio .............................. 10
2.2 Database: Body tracking of the dancer with Kinect ..... 10
2.3 Recording and saving data for a dancing application .. 13
  2.3.1 Project set up .............................................. 13
  2.3.2 Display of body data .................................. 14
  2.3.3 Database recording .................................... 15
3 Dancing with Deep Learning

3.1 RNN

3.2 LSTM

3.2.1 Forget gate

3.2.2 Input gate

3.2.3 Cell state

3.2.4 Output

3.3 MDN

3.4 DWDL: Tensorflow application

3.4.1 Training

3.4.2 Generation

4 Results

4.1 General Results

4.1.1 First test: LSTM

4.1.2 Second test: Introduction of the 75dimensional MDN

4.2 Artistic outcomes

5 Conclusion

5.1 Conclusion

REFERENCES
# LIST OF FIGURES

1.1 Why is a raven like a writing desk? - Gene Kogan .......................... 2
1.2 Pattern Recognition - Memo Akten ........................................ 3
2.1 Skeleton positions relative to the human body ............................. 11
2.2 Example database .................................................................... 13
2.3 Set up: Kinect ......................................................................... 13
2.4 Kinect set up ........................................................................... 14
2.5 Body display ............................................................................. 15
3.1 Unfold Recurrent Neural Network ............................................. 16
3.2 LSTM Network ......................................................................... 18
3.3 Gate example .......................................................................... 18
3.4 Forget gate .............................................................................. 18
3.5 Input gate ................................................................................ 19
3.6 Cell state ................................................................................ 19
3.7 LSTM output ........................................................................... 20
3.8 Mixture Density Network .......................................................... 20
3.9 Mixture Density Network Scheme ............................................ 21
4.1 Recurrent neuronal network: LSTMs ........................................ 26
4.2 Train loss ................................................................................. 26
4.3 Validation loss ......................................................................... 26
4.4 Output Skeleton ....................................................................... 27
4.5 Train loss ................................................................................. 28
4.6 Validation loss ......................................................................... 28
4.7 Output Skeleton ....................................................................... 29
4.8 Train loss ................................................................................. 30
4.9 Validation loss ......................................................................... 30
4.10 Output Skeleton ..................................................................... 30
4.11 Output Skeleton .................................................. 31
4.12 Output Skeleton .................................................. 32
LIST OF TABLES

2.1 JointType Enumeration . . . . . . . . . . . . . . . . . . . . . . . . . . 12
2.2 Joints Properties . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
CHAPTER 1

Introduction

1.1 Thesis definition and goals

This degree thesis follows an approach to machine learning and deep learning disciplines with the goal of obtaining an artistic tool for choreographers and dancers to expand their creations.

When audience visualizes a choreographic piece, they can appreciate the choreographer’s style as well as the feeling expressed by the dancer when performing the movements. Dancing with Deep Learning (DWDL) aspire to create new dance sequences with the same style as the user. The submitted thesis details a recurrent neural network system trained with the dancer’s movement data which provides new sequences of dance. Using this recurrent neural network, the system is capable to understand and generate new choreographies with the same personal style as the user.

The work includes the explanation of the recording of the database, the description of the used algorithm based on a recurrent neural network, the display of the main results, a possible use for artists and future work.

The main goal can be described as follows:

- Expand the inspiration and creation of dance movements for choreographers and performers.

To achieve this goal, we can also propose the following objectives.

- Obtain a database based on the dancer’s movement over time.

- Develop a prototype of recurrent neuronal network for the generation of new dance sequences.

Therefore, starting from the defined goals, they have emerged three subgoals:

1. Create a program to record the data base which will be used to train the neural network. Learn to use a Visual Studio and programming in C# in meaning of recording the multiple hours of the performance of the dancer.
2. Use neural networkings with the help of the Tensorflow library in Python language. To understand the neural networkings and modify them to suit the thesis needs.

3. Present the main results as a visible and easy tool for dancers and choreographers that use Dancing With Deep Learning.

1.2 State of the art

Machine learning has already been used extensively to understand content, as in speech recognition or translation. Nowadays, machine learning is also used to create compelling art and music. Technology has a history of revolutionizing the ways artists create. Some of the works related to art and technology are presented below.

*Why is a raven like a writing desk? - Gene Kogan (2015)*

This work present a reanimation of the tea party scene from Alice in Wonderland (1951), restyled by 17 paintings (Picasso, Frida Kalho…). The project applied the style transfer technique to a scene of the movie (Figure 1.1). The programming code is made by Justin Johnson, based on the paper on style transfer from Gatys, Ecker, and Bethge at the University of Tübingen in Sep 2015.

![Fig. 1.1 Why is a raven like a writing desk? - Gene Kogan](image)

*Machine learning with contemporary dance: Pattern Recognition - Memo Akten (2016)*

A collaboration between choreographer Alexander Whitley and visual artist Memo Akten. Pattern Recognition is a performance for two dancers and light, exploring themes of learning, memory, representation and recognition. The team teach the machine to learn from the dancer’s movements (Figure 1.2). It then recreates those movements later without one of the dancers being present. Light beams are not simply tracking and following the positions of the dancers, but are in fact making real decisions in the moment based on the movements it has learn.
Generative choreography using deep learning - Luka Crnkovic-Friis

They present a deep recurrent neural network model called Chor-rnn. Our work is based on this paper and our goal is to create a LSTM - Mixture Density Network that is able to generate new choreographies.

Generating Sequences With Recurrent Neural Networks - Graves

This paper shows how Long Short-term Memory recurrent neural networks can be used to generate complex sequences with a long-range structure, simply by predicting one data point at a time. They use a 2-dimensional mixture density LSTM. The resulting system is able to generate highly realistic cursive handwriting in a wide variety of styles. For our work, we use the code implemented by Hardmaru (2015) and available at Github, which we have modified for our needs.

1.3 Thesis framework

The project is carried out at the UPC Campus Nord, with the help of the Image Processing Group – BarcelonaTech and teachers Elisa Sayrol and Josep Vidal as a final thesis for a degree in Audio-visual Systems Engineering. The work is based on the paper “Generative choreography using Deep Learning” created by Luka Crnkovic-Friis in which they present a system chor-rnn for generating novel choreographic material from raw motion capture data. The database will be generated entirely by the candidate Gisela Arnal since she has produced the recorded choreographies.

1.4 Methods and procedures

The software development of the project is divided in three main parts. First of all, we use Visual Studio to obtain the database. The code has been written in C# using the
Kinect SDK. On the other hand, the RNN code has been written in Python, using Tensorflow which is a free software library for numerical computation that use data flow graphs. We also use Git and GitHub as our control system version to store the modifications of the code. Finally, to present and save the results of the project, we use a Matlab project.

In order to run and test the main RNN program we have used the cloud servers from the UPC’s Image Processing Group because of the need of resources like GPUs.

1.5 Thesis workplan

The work plan has been divided into six main work packages, which are described in the following tables:
<table>
<thead>
<tr>
<th>Project: Dancing with Deep learning</th>
<th>WP ref: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: bibliographical study</td>
<td>Sheet 1 of 1</td>
</tr>
</tbody>
</table>
| Short description: An analysis of different Deep learning projects in the arts and technology will be done. As a result of this work package, a theoretical foundation for the project development will be obtained. | Planned start date: 28/02/2017
Planned end date: 06/03/2017
Start event: 28/02/2017
End event: 06/03/2017 |
| **T1**: Project proposal, idea conception and Work Plan construction
**T2**: Research some Deep learning and movement related projects | Deliverables: Work plan and raw motion data from Kinect + implementation of some LSTM code. | Dates: |

<table>
<thead>
<tr>
<th>Project: Dancing with Deep learning</th>
<th>WP ref: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Self learning</td>
<td>Sheet 1 of 1</td>
</tr>
</tbody>
</table>
| Short description: A self learning of different programming languages will be done in this work package. | Planned start date: 06/03/2017
Planned end date: 06/04/2017
Start event: 06/03/2017
End event: 06/04/2017 |
| **T1**: VS and Kinect
**T2**: Study of languages for Deep learning algorithm: Tensor Flow. | Deliverables: | Dates: |
<table>
<thead>
<tr>
<th>Project: Dancing with Deep learning</th>
<th>WP ref: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Database recording and VS/C#</td>
<td>Sheet 1 of 1</td>
</tr>
</tbody>
</table>
| Short description: In this work package, I will learn how to use Visual Studio and the Microsoft SDK for Kinect to perform the recording of the project database. Also, a new algorithm to visualize an skeleton with the data obtained through the RNN. | Planned start date: 06/03/2017
Planned end date: 27/03/2017 |
| Dates: | |

<table>
<thead>
<tr>
<th>Project: Dancing with Deep learning</th>
<th>WP ref: 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Mixture density LSTM</td>
<td>Sheet 1 of 1</td>
</tr>
</tbody>
</table>
| Short description: An algorithm as well as some testing will be done in this work package in order to create a Mixture Density Network with LSTMs as RNN. | Planned start date: 03/04/2017
Planned end date: 22/05/2017 |
<p>| T1: Algorithm: programming in tensorflow T2: Test and modifications. | Deliverables: Mixture density LSTM algorithm |
| Dates: | |</p>
<table>
<thead>
<tr>
<th>Project: Dancing with Deep learning</th>
<th>WP ref: 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Dancing with Deep learning, a recurrent neural network to create new choreographies</td>
<td>Sheet 1 of 1</td>
</tr>
<tr>
<td><strong>Short description:</strong></td>
<td>Planned start date: 22/05/2017 Planned end date: 19/06/2017</td>
</tr>
<tr>
<td>In this package, I will perform the training and test of the created algorithm. It is going to be trained with the database that was recorded by the dancer. To visualize the results, a program in matlab is done.</td>
<td>Start event: 22/05/2017 End event: 19/06/2017</td>
</tr>
<tr>
<td><strong>T1: Training and testing of the neural network with the recorded data base.</strong></td>
<td>Deliverables: New choreographies. Program in matlab to visualize the results.</td>
</tr>
<tr>
<td><strong>T2: Presentation of results with matlab.</strong></td>
<td>Dates:</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Project: Dancing with Deep learning</th>
<th>WP ref: 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Documentation and reports</td>
<td>Sheet 1 of 1</td>
</tr>
<tr>
<td><strong>Short description:</strong></td>
<td>Planned start date: 01/03/2017 Planned end date:</td>
</tr>
<tr>
<td>Throughout the entire project, I will perform a documentation of the entire process followed to obtain DWDL.</td>
<td>Start event: 01/03/2017 End event:</td>
</tr>
<tr>
<td></td>
<td>Deliverables: Documentation and reports Dates:</td>
</tr>
</tbody>
</table>
1.6 Deviations from initial plan

The project we present covers the majority of the initial planned tasks. Nonetheless, some deviations from the initial plan had to be introduced. The main reasons were the difficulties we found as we were studying Recurrent Neural Networks.

Results presentation in Matlab: In the initial approach of the project we did not take into account how the results obtained with the RNN would be represented. Then it was suggested to present the results with a program in C# and visual studio to take advantage of the knowledge we had acquired when the database has been recorded. Finally, due to the simplicity, it was decided to create a Matlab program.

Mixture Density LSTM package: It was the most critical work package. Due to the multiple tests without correct results, it was decided at first to only use one Recurrent Neural Networking based in LSTMs. First of all, it was necessary to extend the time spent on the work package of the Mixture Density LSTM. The understanding of the code and the tests slowed down the time planned for this part of the project. Due to the great dimensionality of the data with which the training was done, we had many problems when programming the 75-dimensional MDN. Finally, it was decided to reincorporate the Mixture Density network with LSTM RNN.

1.7 Thesis structure

The following section describes the structure followed by the thesis. In the first place, chapter 1 presents the project introduction and goals. We also find some state of the art references from other artists who have based their works on technology and machine learning. Chapter 2 explains the database acquisition. In it we find a description of both hardware and software used to achieve our goal. In chapter 3, we described the neural network used in our project. First, we make a deep learning introduction as well as an explanation of Recurrent Neural Network, notably LSTM, which are the networks used in our project. Next, we specify the MDNs architectures as well as their use. To close this chapter, the structure, algorithms and development of the actual project are described. In chapter 4, we describe the achieved results over different tests. Finally, chapter 5 summarizes a balance of the results based on the initial project goals and discusses further possible developments to improve "Dancing With Deep Learning".
CHAPTER 2

Kinect for Windows V2

2.1 Specifications

2.1.1 Hardware

Kinect is a line of motion sensing input devices by Microsoft for Xbox 360 and Microsoft Windows PC’s (Kinect 2017). Based around a webcam-style add-on peripheral, it enables users to control and interact with their console/computer without the need for a game controller but through a natural user interface using gestures and spoken commands.

The first-generation Kinect was first introduced in November 2010. A newer version, Kinect 2.0, which is the one that I have used in my work, was released with the Xbox One platform starting in 2013. The Kinect sensor includes a physical device with depth sensing technology, built-in color camera, infrared (IR) emitter, and also a microphone array, can sense the location and movements of people as well as their voices.

Kinect integrates a trio of sensors that work together into this single device:

- An RGB color camera: 640x480 in version 1, 1920x1080 in version 2
- A depth sensor: 320x240 in v1, 512x424 in v2
- An infrared sensor: 512x424 in v2

The Kinect 2.0 face recognition, motion tracking, and resolution are much more precise than the Kinect 1.0. Kinect 2.0 uses “time of flight” technology to determine the features and motion of certain objects. A time-of-flight camera (ToF camera) is a range imaging camera system that resolves distance based on the known speed of light, measuring the time-of-flight of a light signal between the camera and the subject for each point of the image.

The initial version of Kinect allowed us to track up to 20 body joints (Kinect for windows version 2: Body Tracking 2014). The second version allows up to 25 joints. The new joints include the fists and thumbs. Moreover, due to the enhanced depth sensor, the tracking accuracy has been significantly improved and users notice less jittering and
much better stability. For our application, in which we have to detect the tracking of a dancer, we use the second version of the Kinect. With the Kinect sensor we can record the movement of 25 joints at up to 30 frames per second. It can track up to 6 bodies simultaneously but multiple bodies can occlude each other relative to the field of view of the sensor. For my application I just used one body.

2.1.2 Software - Visual Studio

Hardware alone would not make sense without the software. Kinect’s software layer is the essential component to add meaning to what the hardware detects. The Kinect for Windows software development kit (SDK) 2.0 provides data source APIs for Kinect by using C++, C#, Visual Basic, or any other .NET language (Kinect SDK n.d.). I used C++ to create my application. The Kinect for Windows SDK 2.0 includes the following:

- Drivers for using Kinect v2 sensors on a computer running Windows 8 (x64), Windows 8.1 (x64), and Windows Embedded Standard 8 (x64).
- Application programming interfaces (APIs) and device interfaces.
- Code samples.

One of the most important advantages of the Kinect software is that it can distinguish players and their movements even if they’re partially hidden. Kinect extrapolates what the rest of your body is doing as long as it can detect some parts of it. This allows players to jump in front of each other during a game or to stand behind pieces of furniture in the room. Multiple sensors can be used to overcome that problem of occlusion, but it requires more complex software to combine the results. In order to create the Kinect for Windows 2 application, we used Visual Studio (Microsoft Visual Studio 2017). Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. For our application, we are using Visual Studio 2015 and the project is running in Windows 8.1.

2.2 Database: Body tracking of the dancer with Kinect

"Dancing with deep learning" (DWDL) has, as its database, raw motion data of a dancer captured with the Kinect sensor. In other words, the database consists of the movement of the body of the dancer over time. The class that we are using to do the body tracking of the dancer is the Body class, which represents a body. Each body consists of 25
joints. As its name suggest, a joint connects two bones of a skeleton. Figure (2.1) shows the skeleton that is made up of each of these joint types.

![Skeleton positions relative to the human body](image)

**Fig. 2.1** Skeleton positions relative to the human body

In table (2.1) we can see the joint types of a skeleton (*JointType Enumeration* (n.d.)). The Kinect sensor provides us with the position (X, Y, Z) and the rotation information (*JointOrientation*) for each one of them. Moreover, Kinect lets us know whether the joints are tracked, inferred or not tracked. We can see in the table (2.2) the joint’s properties (*Joint Member* (n.d.)).

The *JointOrientation* property get the joints orientations of the body. It is a vector of 4 dimensions (X Y Z W). The *JointOrientation* vector structure represents the orientation quaternion. Quaternions provide a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. The orientation quaternion is the orientation of the parent bone, though for end joints this quaternion is zero, so there is no "roll" angle for the end bones (Joint 3, 15, 19, 21, 22, 23 and 24). In figure (2.2) we can see an example of the coordinates, orientations and states of each joint, which will be used later as a database.

The last column represents the state, that is:

- 0 if joint is not tracked
- 1 if joint is inferred
- 2 if joint is tracked

What we are going to obtain for our database will be the coordinates, orientations and states of the 25 joints of the body of the dancer, at 30 frames per second, over time.
Table 2.1 JointType Enumeration

<table>
<thead>
<tr>
<th>Member</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnkleLeft</td>
<td>14</td>
<td>Left ankle</td>
</tr>
<tr>
<td>AnkleRight</td>
<td>18</td>
<td>Right ankle</td>
</tr>
<tr>
<td>ElbowLeft</td>
<td>5</td>
<td>Left elbow</td>
</tr>
<tr>
<td>ElbowRight</td>
<td>9</td>
<td>Right elbow</td>
</tr>
<tr>
<td>FootLeft</td>
<td>15</td>
<td>Left foot</td>
</tr>
<tr>
<td>FootRight</td>
<td>19</td>
<td>Right foot</td>
</tr>
<tr>
<td>HandLeft</td>
<td>7</td>
<td>Left hand</td>
</tr>
<tr>
<td>HandRight</td>
<td>11</td>
<td>Right hand</td>
</tr>
<tr>
<td>HandTipLeft</td>
<td>21</td>
<td>Tip of the left hand</td>
</tr>
<tr>
<td>HandTipRight</td>
<td>23</td>
<td>Tip of the right hand</td>
</tr>
<tr>
<td>Head</td>
<td>3</td>
<td>Head</td>
</tr>
<tr>
<td>HipLeft</td>
<td>12</td>
<td>Left hip</td>
</tr>
<tr>
<td>HipRight</td>
<td>16</td>
<td>Right hip</td>
</tr>
<tr>
<td>KneeLeft</td>
<td>13</td>
<td>Left knee</td>
</tr>
<tr>
<td>KneeRight</td>
<td>17</td>
<td>Right knee</td>
</tr>
<tr>
<td>Neck</td>
<td>2</td>
<td>Neck</td>
</tr>
<tr>
<td>ShoulderLeft</td>
<td>4</td>
<td>Left shoulder</td>
</tr>
<tr>
<td>ShoulderRight</td>
<td>8</td>
<td>Right shoulder</td>
</tr>
<tr>
<td>SpineBase</td>
<td>0</td>
<td>Base of the spine</td>
</tr>
<tr>
<td>SpineMid</td>
<td>1</td>
<td>Middle of the spine</td>
</tr>
<tr>
<td>SpineShoulder</td>
<td>20</td>
<td>Spine at the shoulder</td>
</tr>
<tr>
<td>ThumbLeft</td>
<td>22</td>
<td>Left thumb</td>
</tr>
<tr>
<td>ThumbRight</td>
<td>24</td>
<td>Right thumb</td>
</tr>
<tr>
<td>WristLeft</td>
<td>6</td>
<td>Left wrist</td>
</tr>
<tr>
<td>WristRight</td>
<td>10</td>
<td>Right wrist</td>
</tr>
</tbody>
</table>

Table 2.2 Joints Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JointType</td>
<td>Gets or sets the joint type.</td>
</tr>
<tr>
<td>Position</td>
<td>Gets or sets the joint position.</td>
</tr>
<tr>
<td>TrackingState</td>
<td>Gets or sets the joint orientation’s tracking state.</td>
</tr>
</tbody>
</table>
2.3 Recording and saving data for a dancing application

2.3.1 Project set up

The main idea of this part of the project is summarized to 1 body and 1 sensor. The objective is to be able to make the recording of the database in any available place, in a simple form and without occlusions in the body of the dancer. The entire database recording was made in the smart room of the D5 building - UPC Campus Nord. Any room with enough space for the dancer to move and without many obstacles between the dancer and the camera would have been accepted.

![Fig. 2.3 Set up: Kinect](image)

**Fig. 2.3 Set up: Kinect**

The procedure followed was, at first, placing the Kinect in front of the dancer at a certain height (e.g. above a table) to capture the entire body of the user (Figure 2.3). We can see an example of how the database was recorded in the figure (2.4). Then, with the first program that allows the capture and visualization of the dancer’s skeleton, different tests were performed. The goal was to see the whole body and to
have a knowledge of the limits in space established by the Kinect. Once the set up was ready, with the help of an external support, the second program was launched, which carried out the capture and presentation of results in a CSV file (example in table (2.2)). The launch of this program had to be done externally since the capture started consecutively. We wanted to avoid erroneous data at the beginning and the end of the recording given the approach of the body to the camera.

![Fig. 2.4 Kinect set up](image)

2.3.2 Display of body data

In order to be able to visualize the dancer’s movement in real time when making the recordings of the database, we implemented a Kinect lab available in Github Kinect for Windows (2015). This laboratory teaches us how to use the body feed from the Kinect to render a skeleton for each of the bodies in a scene and how to use a predefined class to create a skeleton of the body and draw it using simple XAML shapes in a Kinect for Windows 2 application for Windows 8.1.

Since the database recording can be done anywhere, we use this program to be sure that the recording is done correctly and than the sensor captures the correct space. Also, in an artistic interpretation, having a reflection of what the dancer is doing, allows the user to repeat and improve some movements. In figure (2.5) we can see an example of the capture of the body of the dancer during the recording of the database.

Taking into account the constraints of the skeleton, certain movements are not captured correctly. A dissociation of the upper body, a movement centered only on the chest, is not captured by the skeleton, as it has few joints in that area of the body. As we can see, a joint belongs to the neck, one to the center of the body and another directly to the lower part of the pelvis. In this way, if any movement was not correctly captured, the dancer could visualize it in real time and change it progressively.
2.3.3 Database recording

To obtain the database as coordinates stored in a CSV we use an application from Sebastien Andary (2016). This program gives us more information than we need, so we will have to filter the data later. Figure 2.2 shows us an example of the output of this program.

The program is capable of capturing up to six skeletons, but only saves the information of the first one it sees.
Deep learning is part of the family of machine learning methods based on learning data representations. It allows computational models that are composed of multiple layers to learn representations of data with multiple levels of abstraction. Deep learning discovers structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. The methods are “representation-learning” with multiple levels of representation. They are obtained by composing simple modules that transform the representation at one level (starting with the input) into a representation at a higher, slightly more abstract level. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains (LeCun (2015)).

3.1 RNN

The main idea behind RNNs is to make use of sequential information. RNNs process an input sequence one element at a time, maintaining in their hidden units a ”state vector” that implicitly contains information about the history of all the past elements of the sequence. We can think that RNNs have a “memory” which captures information about what has been seen so far.

![Unfold Recurrent Neural Network](image-url)
In figure (3.1) we can see an unfold Recurrent Neural Network. A RNN can map an input sequence with elements $x_t$ into an output sequence with elements $o_t$, with each $o_t$ depending on all the previous $x'_{t'}$ (for $t' < t$). In this figure, $x_t$ is the input at time step t. $s_t$ is the hidden state at time step t. It’s the “memory” of the network. $s_t$ is calculated based on the previous hidden state and the input at the current step. $o_t$ is the output at step t. RNN equations can be found in (3.1).

$$s_t = \tanh(Ux_t + Ws_{t-1}) \quad o_t = \text{softmax}(Vs_t)$$

(3.1)

To train RNNs we use Back Propagation Through Time (BPTT) which is a training algorithm used to update the network weights. The goal is to calculate the gradients of the error with respect to the parameters $U$, $V$ and $W$ and then learn good parameters. To calculate these gradients we use the chain rule of differentiation. Because $W$ is used in every step up to the output we care about, we need to backpropagate gradients till the beginning of the network. Gradient contributions from “far away” steps become zero (Razvan Pascanu 2012) and the state at those steps doesn’t contribute to what we are learning: we end up not learning long-range dependencies. RNNs have difficulties learning long-range dependencies, this is called the Vanishing Gradient Problem.

### 3.2 LSTM

The Vanishing Gradient Problem prevents standard RNNs from learning long-term dependencies. LSTMs were introduced by Hochreiter and Schmidhuber (1997), and are explicitly designed to avoid the long-term dependency problem that RNN have through a gating mechanism. A LSTM network is an artificial neural network that contains LSTM units instead of other network units (Oinkina (2015)). A LSTM unit is a recurrent network unit that excels at remembering values for either long or short durations of time.

As current RNN, LSTMs also have the chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting between them. We can see in the figure (3.2) an example of LSTM network. Each line carries an entire vector, from the output of one node to the inputs of others.

The LSTM has the ability to add or remove information regulated by structures called gates. Gates are a way to optionally let some information through. They are composed out of a sigmoid neural net layer and a point wise multiplication operation as we can see in Figure (3.3).

Their name is "gate" because the sigmoid function squashes the values of the vec-
tors between 0 and 1, and by multiplying them by another vector we choose how much of that other vector we want to ”let through”.

3.2.1 Forget gate

First of all we have the forget gate which defines how much of the previous state we want to let through and what information we are going to throw away(Figure 3.4). It looks at $h_{t-1}$ and $x_t$, and outputs a number between 0 and 1 for each number in the cell state $C_{t-1}$. A 1 represents ”completely keep this” while a 0 represents ”completely get rid of this.”

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

3.2.2 Input gate

The input gate defines how much of the newly computed state for the current input we want to let through (Figure 3.5).
3.2.3 Cell state

The horizontal line running through the top of the diagram represents the cell state, which is the core of a LSTM. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged. Intuitively the cell state is a combination of how we want to combine previous memory and the new input. To update the cell state, we multiply the old state by $f_t$, forgetting the things we decided to forget earlier. Then we add $i_t \cdot C_t$. This is the new candidate values, scaled by how much we decided to update each state value (Figure 3.6).

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

3.2.4 Output

Finally, we need to obtain what we are going to output. Given the memory, we compute the output hidden state by multiplying the memory with the output gate. We first run a sigmoid layer through the cell state and then, we put the cell state through $\tanh$ (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to (Figure 3.7).

We can say that the gating mechanism is what allows LSTMs to explicitly model long-term dependencies. By learning the parameters for its gates, the network learns how its memory should behave (Denny Britz (2015)).
3.3 MDN

Mixture density networks (MDN) (Bishop (1994), 1994) are a class of models obtained by combining a conventional neural network with a mixture density model.

The idea is to use the outputs of a neural network to parameterize a mixture distribution. A subset of the outputs are used to define the mixture weights ($\pi$), while the remaining outputs are used to parameterize the individual mixture components: means ($\mu$) and variances ($\sigma$). The mixture weight outputs are normalized with a softmax function to ensure they form a valid discrete distribution, and variances are passed through an exponential function, to ensure they are defined positive. When used with RNN, the output distribution is conditioned not only on the current input, but on the history of previous inputs.

The probability density of the target data is then represented as a linear combination of functions in the form:

$$p(t|x) = \sum_{i=1}^{m} \pi_i(x) \Phi(t|x)$$

(3.2)
where \( m \) is the number of components in the mixture. The functions \( \Phi(t|x) \) represent the conditional density of the target vector \( t \). Various choices for the \( \Phi \) functions are possible. However, the most used are Gaussian of the form:

\[
\Phi(t|x) = \frac{1}{\sigma(x)\sqrt{2\pi}} e^{-\frac{(t-\mu(x))^2}{2\sigma(x)^2}} \tag{3.3}
\]

It is also possible to introduce full covariance matrices for each Gaussian mixture, at the expense of a more complex computation, specially if the dimensionality of the input is high. Thus, the representation given by (3.1) and (3.2) is completely general, in the sense that it does not assume that the components of \( t \) are statistically independent.

Mixture density networks are trained by minimizing the negative log probability density (error function) of the targets under the induced distributions. In order to minimize the error function, we need to calculate the derivatives of the error with respect to the weights in the neural network. For that purpose we can use a gradient descent optimization algorithm (one of the most popular algorithms to perform optimization and the most common way to optimize neural networks) like Adam Optimizer.

### 3.4 DWDL: Tensorflow application

At the core of dancing with deep learning we found a LSTM - Mixture Density Model. The code has been implemented in python using the Tensorflow library. (An open-source software library for Machine Intelligence). The DWDL implementation is performed with a mixture density network. As explained at section 3.3, at the output of the neural network we obtain a parameter vector which is going to be mapped to a mixture of distributions (Figure 3.9). The parameters we want to predict are \( k \) mixture components \( \pi \) (weights) and the \( \mu \) (mean) and \( \sigma \) shape parameters for each of the \( k \) gaussians.

![Fig. 3.9 Mixture Density Network Scheme](image-url)
3.4.1 Training

As we have seen in section 2.3, our database is composed of many parameters. We will filter the CSV file obtained to keep the 3 coordinates (X, Y and Z) of the 25 joints of the dancer’s body which makes a total of 75 coordinates. The input of the MDN is going to be a vector of sequence length equal to 90 of the 75 coordinates at each time step (3 seconds of dance at 30fps).

In our test, we used a 2-layer basic-LSTM network with 256 nodes in each layer.

```python
cell_fn = tf.nn.rnn_cell.BasicLSTMCell
cell = cell_fn(args.rnn_size, state_is_tuple=False)
cell = tf.nn.rnn_cell.MultiRNNCell(
    [cell] * args.num_layers,
    state_is_tuple=False)
```

Our input data is 75-dimensional and we used 25 75-dimensional gaussians for the mixture model. Our goal is to model the conditional distribution as a mixture of Gaussians, where each Gaussian component parameters are dependent on the input, i.e:

$$
P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$  \hspace{1cm} (3.4)

According to our model, the Recurrent Neural Network has 3 types of outputs: the mixing coefficient (weights) \( \pi \), means of the Gaussians components \( \mu \), and their variance \( \sigma \). The mixing coefficients and variances are subject to constraints (section 3.4).

$$\sum_{n=1}^{25} \pi = 1 \hspace{1cm} \sigma > 0$$  \hspace{1cm} (3.5)

We have 25 Gaussians so we will need a total of \( NOUT = self.num_mixture * 151 = 3775 \) outputs that represent the weight parameters of Gaussians (1-dimensional), means (75-dimensional) and variances (75-dimensional).

```python
z = output
M = self.num_mixture
z_pi = z[:, 0: M]
z_mu = z[:, M: 76 * M]
z_sigma = z[:, 76 * M: 151 * M]
```

The first 25 outputs are mapped to the 25 mixing weights \( \pi \) by putting them through a softmax activation function (equation 3.5), this ensures they are positive and add to one.

$$\pi_i = \frac{exp(z_i^\pi(x))}{\sum_j exp(z_j^\pi(x))}$$  \hspace{1cm} (3.6)
The next $75 \times 25$ are mapped to the mean and the last $75 \times 25$ to the variance. For the variance we take the exponential (equation 3.6) of those so we can guarantee that they are positive as required.

$$\sigma_i = \exp(z_i^\sigma)$$  \hspace{1cm} (3.7)

For regression problems the squared error is usually minimized, in this case however we minimize the negative log likelihood of the data.

$$\text{result} = -\text{tf.log(result)}$$

3.4.2 Generation

After the training, our network can generate samples and save them in a CSV file. These samples represent the 75 coordinates of the dancer’s body through time. First of all we start by emptying the states of the LSTM network, and passing into the network an initial input. This input can be a random input or some initial position of the body dancer.

```
prev_x = np.random.rand(1, 1, 3 * 25)
prev_state = sess.run(self.cell.zero_state(1, tf.float32))
```

After the initial input, a zero state is passed into the network with the trained model. We then get a set of parameters from the output of the network, that are going to be the parameters of the 75-dimensional mixture distribution that defines the probability distribution of where the next position of the body will be located. We randomly sample a set of values from this distribution, and then we save the coordinates to a CSV file. Afterwards, we repeat this loop, feeding in the output of the network as new inputs.

```
for i in range(num):

    feed = {self.input_data: prev_x, self.state_in: prev_state}
    [o_pi, o_mu, o_sigma, next_state] = sess.run(
        [self.pi, self.mu, self.sigma, self.state_out], feed)
    idx = get_pi_idx(random.random(), o_pi[0])
    next_x1 = sample_gaussian_2d(o_mu[0, :, idx],
        o_sigma[0, :, idx])
    dance[i, :] = next_x1
    params = [o_pi[0], o_mu[0], o_sigma[0]]
    mixture_params.append(params)
```
prev_x = np.zeros((1, 1, 3 * 25), dtype=np.float32)
prev_x[0, 0:] = np.array([next_x1], dtype=np.float32)
prev_state = next_state
CHAPTER 4

Results

4.1 General Results

The database recording was done during different days. The dancer recorded sequences of 30-40 minutes of choreography. Therefore it is necessary to take into account that every 30 minutes approximately, there is a sudden change of the position of the skeleton.

The results at the output of the recurrent neural network were presented with a small Matlab project, which showed the skeleton resulting from Dancing with Deep Learning. The output is saved on a CSV text format and therefore visualized in Matlab. The train_loss and validation_loss graphs were obtained with Tensorboard. TensorBoard operates by reading TensorFlow events files, which contain summary data that is generated when running TensorFlow. It is the visualization tool that can be used to visualize TensorFlow’s graph and plot quantitative metrics about the execution of our graph. These graphs show the evolution of train and validation loss in relation to the number of batch used in the training. The number of batch is calculated when performing the train of the network.

4.1.1 First test: LSTM

In the paper “Generative choreography using deep learning” Luka Crnkovic-Friis (2016) it is explain that:

In the case of continuous data, [...] it can be shown that when using a mean square error metric, the output will stagnate and converge to an average output (Bishop, 1994).

The first test was performed using only a recurrent neural network composed of LSTMs. It is composed of two hidden layers of 256 LSTM (figure 4.1). Both values can be modified during the train phase of the neural network.

100 epochs: 2 layers of 256 LSTM
Fig. 4.1 Recurrent neuronal network: LSTMs

Validation loss is the error after running the validation set of data through the trained network. Usually, as the epochs increase both validation and training error drop. At a certain point, while the training error continues to drop (the network learns the data better and better) the validation error begins to rise. This effect is called over-fitting. We can’t see this effect in figures (4.2 - 4.3).
It is seen that the values of both graphs converge from approximately 40 epochs. Even so, when we perform the skeleton representation in Matlab we observe how the effect described at Luka Crnkovic-Friis paper takes place, within milliseconds, the skeleton is stagnant and remains immobile (Figure 4.4).

1 epoch: 2 layers of 256 LSTM

We can see in figure 4.5 and 4.6 the results of the following test.

If we only trained our neural networking with one epoch and the five hours recorded, we obtain the next result. We can appreciate how the skeleton does not have a clear shape: the body of the dancer is not distinguishable (Gollum effect). We still have the same effect of body paralysis (Figure 4.7).

4.1.2 Second test: Introduction of the 75-dimensional MDN

Unlike the work done by Graves which one it is used MDN’s of two dimensions due to on line writing generated with the coordinates X and Y for the representation, we have to work with a 75 value dimensionality: 25 joints and (X Y Z) coordinates per joint.

After many tests, it is decided to do the MDN with 75-dimensions gaussians. To make this process, we used the procedure explained at the point 3.3 MDN. At the output of our LSTM recurrent neural networking, we get the parameters that with we will use to calibrate our mixture density block. To achieve this, we make a partition of the LSTM output, to obtain the Gaussian weight, means and sigmas. Since we have a high dimensionality, we use diagonal covariances.

We observed in both train loss (figure 4.8) and the validation loss (figure 4.9) a big
fluctuation of values.

When we make the generation of our DWDL, we initialize the networking with 75 random values, which it represents the initial dancer body’s values. The generated result we get as output of our neural networking is shown in figure 4.10. We observe that the skeleton is not displayed.

The following test it is done with an initialization of non random values. The generation of the DWDL is initialized with a skeleton in a neutral position. It is seen that we have the same behavior as the initialization with random values. We finally we decided to take as a value of the gaussians the means we obtained. We get the result shown in figure 4.11. When we generate our output using random Gaussians with the means and variances obtained when training, we don’t observe a skeleton. When only using the means as outputs, we obtain a skeleton.

Even though it is a meaningful skeleton, it does not move consistently and does not dance like the style we are looking for. The movements are sudden (the body moves from one region to another) and sometimes we even lose the skeleton shape. We can say that we have obtained a basic result for the creation of choreographies. Additional
works could complete the results we have obtained in order to obtain the dance sequences we are looking for.
Fig. 4.8 Train loss

Fig. 4.9 Validation loss

Fig. 4.10 Output Skeleton
Fig. 4.11 Output Skeleton
4.2 Artistic outcomes

Humans and not humans. Art in the age of Artificial Intelligence by Google - AMI. Program presented at the Sonar+D 2017 in Barcelona. The project could be presented within the AMI as a tool for dancers and choreographers. In AMI, works are developed together alongside artist’s current practices and shown at galleries, biennials, festivals, or online.

Machine/Human Collaboration. In front of a possible lack of choreographic material, the dancer can use Dancing With Deep Learning to inspire himself when creating new compositions. Likewise, with these new compositions the machine can train again and so progressively.

Fig. 4.12 Output Skeleton

It can be compared working with Dancing With Deep Learning as a virtual partner, which has the same choreographic language, that is, the same style of the dancer who uses it. The out coming work it could be presented inside a dance piece with either the material obtained for the dancer, the DWDL material or as a set of both materials, the one of the dancer and the DWDL one.
CHAPTER 5

Conclusion

5.1 Conclusion

As a conclusion we can say that some promising preliminary results have been obtained to build an automatic tool for creating new choreographies. Recording the database on the dancer’s movement over time was successfully achieved. The data was saved in CSV files and was used to generate videos making use of a Matlab program to show the moving skeleton. On the other hand, we could not entirely generate a dancing skeleton at the output of DWDL with a meaningful new choreography. Two of the reasons why we believe it has not been possible to obtain the expected results are:

- At the end of the Mixture Density Network, mixtures are chose in a random way.

- Given the great dimensionality of our data (75-dimensional), we only could use diagonal covariance matrices, which represent the variance of the data. This implies that we only need 75 coefficients to represent the covariance matrix instead of \((75 \times 75/2 \approx 2813)\) coefficients. That’s way we are not taking into account the relation between joints in a body (For example if one hand is raised, the elbow may also have a similar movement).

DWDL it not yet a functional tool in the artistic field, but it’s a preliminary tool towards the creation of new choreographies.

Future development:

As a future development, first we should modify the recurrent neural networking to obtain meaningful results. Although some tests have been performed, the MDN-Recurrent NN needs further analysis and tuning of the parameters. On the other hand, we also want to extend the input data to include music. In that way, DWDL will generate new choreographies related to a musical composition. Furthermore, we could also think about using not only the skeletons but also the video images. In this respect, we
could get inspiration from Graves’ paper, where input text is related to synthetic written text.

Dancing with deep learning uses for the database recording, one sensor and one professional dancer. We also thought it might be interesting to use multiple bodies to create a choreographic composition represented by more than one person so the dancers could interact.
REFERENCES


