A Variational Autoencoder Approach for Representation and Transformation of Sounds

- A Deep Learning approach to study the latent representation of sounds and to generate new audio samples -

Master Thesis

Matteo Lionello

Aalborg University Copenhagen
Department of Architecture, Design Media Technology
Faculty of IT and Design

The current Master Thesis report has been written in LaTeX, the images in the Results and Methods chapters have been rendered by using Python and Matlab programming languages. Waveform screens are taken from Audacity software and the graphs have been created by using Cacoo, an online graph render: https://cacoo.com/

AALBORG UNIVERSITY
STUDENT REPORT

Faculty of IT and Design,
Aalborg University Copenhagen

Author: Matteo Lionello

Supervisor: Dr. Hendrik Purwins

Page Numbers: 55

Date of Completion: May 31, 2018

Abstract

Deep Learning introduces new way to handle audio data. In the last few years, audio synthesis domain started to receive many contributions in sound generation and simulation of realistic speech from the Deep Learning field. Specifically, a model based on a Variational Autoencoder is hereby introduced. This model can reconstruct an audio signal 4096 samples long (512ms with 8KHz sample rate) after a $\times 32$ compression factor from which 20 features have been collected to describe the whole audio. The 20 features are then used to reconstruct back the input. The model is adapted to collect items of the same class in the same sub-region of the compressed features space. This approach allows morphing operations between sub-regions corresponding to different types of audio. In order to reach this goal two loss functions are used in the model: a reconstruction loss regularised with a variational loss and a classification loss applied at the bottleneck. This report compares the results of three Variational Autoencoders based on convolution layers, dilation layers and a combination of them. The evaluation of the models is based on 1-Nearest Neighbour classification at the bottleneck and a Dynamic Time Warping based on MFCC features to evaluate respectively the instances distribution in the bottleneck layers and the goodness of the reconstruction task. Finally a music application is introduced for drum samples.

AALBORG UNIVERSITY
STUDENT REPORT

Fakultet for IT og Design,
Aalborg Universitet København

Ophavsmand: Matteo Lionello

Tilsynsførende: Dr. Hendrik Purwins

Sidetal: 55

Dato for Afslutning: May 31, 2018

Abstrakt

Deep Learning introducere en ny måde at håndtere lyddata på. For nyligt begyndte domænet indenfor lydsyntese at modtage mange bidrag der omhandler lydgenerering og simulering af realistisk tale ved hjælp af Deep Learning. Et tilfælde er en model baseret på Variational Autoencoder. Denne model kan konstruere et lydsignal på 4096 samples (512ms med 8kHz samplings frekvens) efter en $\times 32$ ganges kompressionsfaktor, hvoraf 20 funktioner er blevet samlted for at beskrive hele lyden. De 20 funktioner bruges derefter til at rekonstruere inputtet. Modellen er bygget til at samle elementer af samme klasse i samme underområde af det komprimerede element. Denne fremgangsmåde tillader morphing operationer mellem underregioner svarende til forskellige typer lyd. For at nå dette mål anvendes følgende funktioner i modellen: et genopbygningsforløb reguleret med afariationstab og et klassificerings tab anvendt på flaskehalsen. Denne rapport sammenligner resultaterne af tre Variational Autoenvoders baseret på convolution lag, dilation lag og en kombination af dem. Evalueringen af modellerne er baseret på 1-Nearest-Neighbour klassifikation ved falskehalsen og en Dynamic Time Warping baseret på MFCC funktioner for at evaluere henholdsvis forekomstfordelingen i modellens lag og kvaliteten af genopbygningen. Til sidst introduceres en musikapplikation til trommeprøver.

Acknowledgements

I would like first to thank my supervisor Dr. Hendrik Purwins, who patiently guided and advised me during my studies for most of the semester projects I had and for hereby thesis. I am also grateful to all the staff of Sound and Music Computing for their work to realise this project-research based Master study. I would also acknowledge my bachelor thesis co-supervisor Dr. Marcella Mandanici, who first introduced me to Sound and Music Computing world and suggested me to start this Master. I want to express my gratitude to my parents and to my family who always gave me wise advise besides moral and economic support during my whole study career. I am grateful to all the friends of mine, either local and abroad, who gave me help and support across the years of my Master. A special thanks goes to Paolo, Laura, Alicia and Jonas for their advise during the writing of the thesis. Finally I would like to thank the person without who anything of this would have been possible, myself.

Contents

Preface

The current report describes the work done at Aalborg University during the Spring semester 2018 in relation to the Master Thesis in Sound and Music Computing. The work, supervised by Hendrik Purwins, has been conducted during the final term of the studies of the Master and it takes place in a machine learning context. The interest in machine learning in musical and sound contexts is motivated by the recent evolution of machine learning tools and applications.

The thesis cover a 3 month-period, during which most of the time has been spent studying and developing program codes.

The project has brought multiple challenges. Because of the innovative strategy in audio context proposed in the thesis, the whole code has been written from scratch and the Python library "Tensorflow" has been learnt during the project. The project deals with complex structures and many parameters. Understanding the contribution of each structure was part of the goal of this thesis, including learning which combinations of these provided a good strategy to solve the problems encountered.

The thesis topic has been decided according to the previous academical projects conducted during the Master. All semester projects were, indeed, chosen in machine learning context, investigating long short-term memory models, parametric t-distributed stochastic neighbour embedding and modelling expressiveness of vibrato of violin performance in a machine learning approach.

Aalborg University, May 31, 2018

Matteo Lionello
<mlione16@student.aau.dk>

Chapter 1

Introduction

Artificial Intelligence sectors are forcefully growing up. Their application covers inter-disciplinary research fields such as economics, biology, medicine, telecommunication, automata, arts and videogames. Autonomous cars, speech recognition and synthesis, virtual personal assistant, emails filtering, video surveillance, optimization of search engine browsers, social media services, online customer support, recommendation systems and so on, are some examples of daily human's interactions with AI systems by means of Deep Learning methods.

In the last decade, there has been a growth in Deep Learning techniques implementation in the images domain whose main goals are classification, transformation and generation of images. Besides similarities between images and audio data, the amount of studies regarding Deep Learning in audio contexts is less than its applications in the image domain. Adapting the same algorithm to work in both domain could be not such an easy task because of the large differences between images and sound instances. Deep Learning is usually used in audio domain for feature extraction, music information retrieval [47, 46, 10], acoustic event classification[56, 49] and, in the last couple of years, audio transformation and synthesis.

Audio synthesis comprehends sets of methods to generate sounds usually from scratch. It finds its main applications in music production - synthesizers - and text-to-speech generation - vocoders. There exist multiple strategies to create sounds to emulate existing music instruments or to create new sounds of instruments that actually do not exist in the real world. The main techniques consist of granular sythesis[54], frequency modulation[6], subtracting synthesis, additive synthesis, sampled synthesis, wavetabels[2], modelling physical systems. On the other hand, speech synthesis is usually done by formant synthesis, concatenative synthesis or statistical parametric synthesis. Research in artificial multi-layers neural networks, has provided new tools to generate sound for either speech and sound synthesis[9, 50].

In this scenario, one interesting topic of ML concerns generative models which are capable to create new data with respect to what used during the learning process (see section 2.2.3). These models have being largely used in computer vision to increase the resolution of the images, to delete or add objects appearing in it, to change some visual properties, to reduce noise, or to generate completely new images. In audio domain generative models have been mainly investigated by Google Magenta and Goodle Deep Mind research teams. These researches have investigated both speech and sound of musical instrument synthesis by abstracting and then recomposing back one object (e.i. one audio file) in a stack of hierarchical representations of it (see section 2.1). One interesting idea is how to interact with one of this representations to reconstruct an object that will be slightly different from the original one. The idea is to create a model which learns to manage a large set of objects, mapping their attributes in an *abstract representation* by creating unique descriptions of the objects. This method extrapolates a small amount of important features which allow the correct reconstruction of the object.

Let us imagine a potter who needs to build a copy of an existing vase. Let us assume that s/he can look at the original object only before and after the production of the copy. After having looked at it for a while, s/he will have a first impression of the object in her/his mind and then s/he will try to reconstruct it. Once the production process ends, the potter will look back to the original vase and compare her/his product with the target object. By looking at the differences between the two objects, the abstract impression in her/his mind improves, in order to produce smaller dissimilarities the next time s/he will have to reconstruct the object. At this point we should distinguish a memory based representation, where the abstract impression is an approximation of the real object, from a different structure which could be a list of information and properties of the real object[1]. Every time this process happens, the potter will learn a *better* representation of the object in her/his mind which allows her/him to improve the copy of the vase. In this example, the process happens in three different spaces: the 3-dimensional original vase, the abstract representation inside the potter's mind and the 3-dimensional reconstructed pottery. Now, let us suppose there exists a joker-homunculus inside the potter's mind who is able to interact with the abstract representation; once the potter finishes to *learn* a good way to rebuilt the original objects, the homunculus inside her/his is able to change the structure of the abstract representation in order to change how the potter will build the vase. By changing some properties of this representation, the homunculus will be able to control the potter's acquired knowledge to build new vases, slightly different from the original one used for the potter's training. In this example, the homunculus introduced is nothing but the capability of imagination which leads the person to create new things starting

[1]This concept will be considered later and explained in section 4.2. The idea is to distinguish a model which conserve a fragment sort information of the original data, rather than to extract the essential features sufficient to describe uniquely the original object.

from what s/he learns.

Similarly to the previous example, the goal of this project is to create a model for generating new sounds that can be easily controlled by changing the intermediate representation of a model. The model is built to learn to reconstruct a set of audios by extracting a low-dimension representation from them. In the current thesis, this is achieved by means of Deep Learning methods based on autoencoders which *learn* how to represent a given dataset by extracting the features necessary to reconstruct it. The feature extraction is done by the autoencoder by projecting the audio into a low-dimension space defined as a probability space. Once the training process of the model ends, it makes it possible to explore the probability space and to create from it a new representation between two regions that collect two different kind of sounds. By reconstructing the set of points sampled from a path starting from one region and ending in the other one, a morphing between the two sounds will be achieved. The project gathers multiple methods of deep learning techniques such as autoencoders (AE), variational autoencoders (VA-AE) and future implementation of generative adversarial networks (GAN). The project investigates these research fields, implementing these approaches with audio data.

The thesis is structured as follows:

- **Chapter 2**: Related Works. A description of historical context of deep learning is proposed introducing its concept according to its time-line. Further, this chapter contains a description of the projects that provides the state of the art and of other academic projects to contextualise the environment in which the current thesis is done.
- **Chapter 3**: Probability data representation in Machine Learning context. It contains a formal introduction of Autoencoders and Variational Autoencoders.
- **Chapter 4**: Experiments. It consists in a description of some preliminary experiments conducted and the methodologies used.
- **Chapter 5**: Results. It provides a description of the dataset used and the results obtained from the models introduced in the previous chapters.
- **Chapter 6**: Example of Application in Musical Context.
- **Chapter 7**: Conclusions and Future Work.

Chapter 2

Related Work

This master thesis originates from a context of successful results obtained by particular methods of Machine Learning applied to sound synthesis and picture generation. **Machine Learning** (ML) is the technique used in Artificial Intelligence to give the machine the ability to "*learn*". The learning process consists in improving the ability of making predictions by adapting parameters of statistical models to a particular task. **Deep Learning** is a particular ML framework that creates a hierarchical representation map to extract abstract and hidden concepts from the knowledge of the data used in a model. One goal of Deep Learning also regards to understanding what these hidden concepts refer to and how to interact with them. In the last couple of years, a rapid evolution of methods to generate data by means of Deep Learning techniques has been observed, providing interesting perspectives in how to represent and generate audio and pictures. As seen by the large amount of projects which are based on it described in section 2.2, the most relevant work achieved in audio domain by means of Deep Learning tools is the introduction of WaveNet model [50] (see sec. 2.2.1) by Google DeepMind team[1]. In this section is proposed a brief historical excursus of how we arrived to deep learning and state of the art explaining, for each step, basic neural network elements and algorithms.

2.1 What is Deep Learning?

Deep Learning is a branch of Machine Learning that models the data through a multi-level representation. Each level of the model is associated to a hierarchical representation of the data. One instance of Deep Learning method is the multi-layer Perceptron, also known as deep feed-forward neural network (DNN). The history behind of the artificial neural network is strictly linked to the studies of neural behaviour of the brain.

[1] `https://deepmind.com/`

The birth of Artificial Intelligence is given by an extraordinary meeting of different research fields. In fact, it is strictly linked to computational biology, automata theories, psychology and logic. During the last decade of the first half of the Twentieth Century, these research fields converged to a common research area which is nowadays called computational neuroscience. Even if the term artificial intelligence was coined only in the 1955[27], the idea behind it has accompanied humans throughout all their history. The possibility to create machines that could compete with human kind or that can simulate their action, behaviour and smartness, has always being a point of interest in human's mind, across all the historical eras. In this context the most remarkable formal point in the history is of course the Darthmouth Conference in 1956[27], for which occasion the term "Artificial Intelligence" was first coined by McCarthy. To reach that moment in the history we need to go back at least of 65 years before when William James, at the end of the 19th Century published a book called "*Principle of Psychology*"[17]. In this book he first stated how brain could work in term of **inter-connections** of points. Particularly he suggested that the activity of the brain in one point is the results of the sum of the activity in the neighbour points which discharge their energy into it.

More than 50 years after, two researchers, McCulloch and Pitts[28], derived the model of a neural systems according to the biological knowledge at that time. In their book they proved that any boolean function could be represented by networks composed by their neurons. Their model of neuron has a similar structure to the actual artificial neuron we use today, summing together weighted inputs and passed them to a transfer function (compare to the elements in figure 2.2). However, there occur some dissimilarities respect to the current model. For example, one of the assumption their model relied on was that the neuron is a all-or-none process of binary inputs (excitatory and inhibitory synapses). They also proposed that the phenomena of learning was given by permanent alteration in the structure of the net. Because of that they used a threshold element as transfer function. For this reasons this model is usually referred as **Threshold Logic Unit** (TLU). The current learning concept on which neural networks are based was described in 1949. In that year Donald Hebb published "Organization of Behavior". In this work he proposed how neurons adapt their synaptic weights during the learning processes. The mechanism of the synaptic plasticity has been referred as **Hebbian learning** and its the today idea of how a neural network update its weight during the training process.

> "... The alternative approach, which stems from the tradition of British empiricism, hazards the guess that the images of stimuli may never really be recorded at all, and that the central nervous system simply acts as an intricate switching network, where retention takes the form of new connections, or pathways, between centers of activity." - F. Rosenblatt, "The Perceptron" 1958

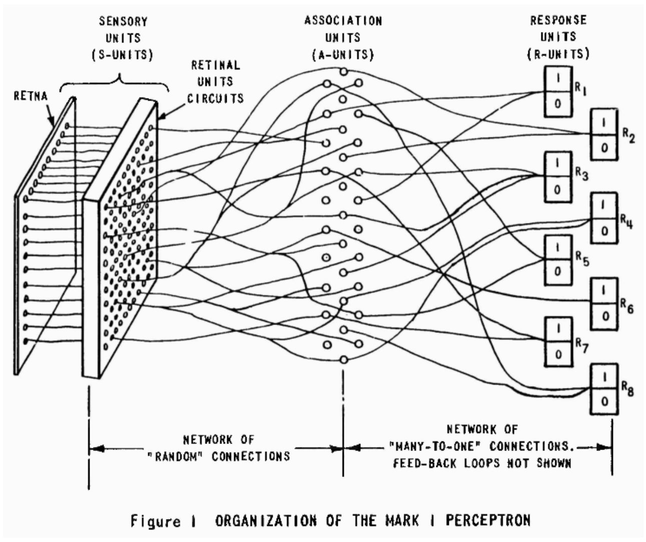

SENSORY UNITS (S-UNITS)
ASSOCIATION UNITS (A-UNITS)
RESPONSE UNITS (R-UNITS)
RETINA
RETINAL UNITS CIRCUITS
R1
R2
R3
R4
R5
R6
R7
R8
NETWORK OF "RANDOM" CONNECTIONS
NETWORK OF "MANY-TO-ONE" CONNECTIONS. FEED-BACK LOOPS NOT SHOWN
Figure I ORGANIZATION OF THE MARK I PERCEPTRON

Figure 2.1: Schematic representation of Mark 1, the first implementation of a perceptron. Image taken from the "MARK I PERCEPTRON OPERATORS' MANUAL" `http://www.dtic.mil/dtic/tr/fulltext/u2/236965.pdf`

In the late '50s at the Cornell Aeronautical Laboratory, Frank Rosenblatt worked on a machine named Mark 1[43]. The machine implemented a **perceptron** network. Compared to McCulloch and Pitts's TLU, Rosenblatt proposed a new model which was able to learn the weights and the thresholds values. The machine was designed to learn shape classification. It consisted (see figure 2.1) of a 400 matrix of photocells which send the electrical impulse to a set of 512 association cells in a projection area. Both stimulus and response are still assumed as all-or-nothing component. Then some random connections take place between the projection area and a second area called association area, counting 40 units. Each units in this second area receive a certain amount of connections from the first one. These units are structured as TLU whose input were summed together and then compared to a threshold. 8 response units were bidirectionally connected to the association units in the association area. The model was able to reach good results in discriminate different image stimuli illuminated over the photocells area by automatically changing the the values of the potentiometers in the connection between the units in the association area and the respective response units. The automatic change of the potentiometers was done by electrical motors.

Adaline[52] is a similar machine of reduced physical dimension than Mark 1. It works similarly to the previous machine. It improved the learning method by introducing the least mean square (LMS) algorithm which is based on the **stochastic gradient descend** method commonly used today in neural networks.

In 1969 Marvin Minsky and Seymour Papert published a book called "*Perceptron*" collecting some analysis and studies in the field. In this book they wrote how

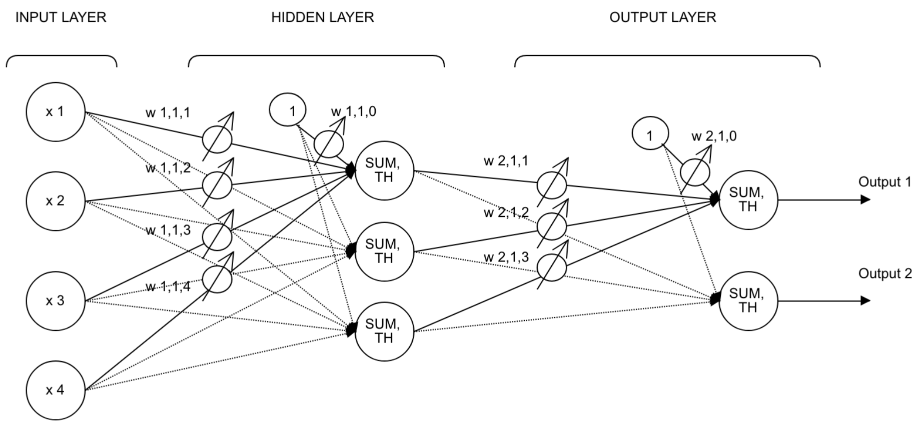


Figure 2.2: Example of a 3 layers perceptron network. By comparison to Mark 1 and Adaline machines, here is introduced an hidden layer between the input and the output layers. The internal nodes are the circle and "SUM, th" refers to the neurons of the network performing a sum operation of their input and then passing the result through a threshold element. Each of them is actually a TLU neuron. The potentiometers (circles with arrow) refer to the weights which are changed (automatically in the Mark 1, and manually in the Adaline) during the tuning.

perceptrons and their possible extensions were a sterile field. Their conclusions directly affected the amount founding in the research field and here started a poor period of the artificial neural network history.

2.1.1 XOR Problem

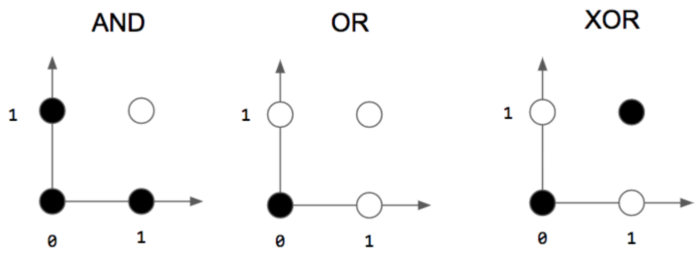


Figure 2.3: Perceptrons consisting of just one single layer of affected by weights (excluding so from the figure 2.2 the hidden layer) are able to compute linear discrimination (AND and OR operations), but in the case of the exclusive or (XOR) it is impossible to teach the network to discriminate correctly these three regions in 2 classes.

The problem raised by Mynsky and Papert included specifically the XOR problem. It is, in fact, impossible for one perceptron to process a exclusive-or problem (see figure 2.3). Nowadays we know that this problem can be solved by a **multi-layer**

perceptron (see figure 2.2). By adding a second layer, a neural network is indeed capable to solve non-linear separation. Trying to find the right way to set the weights in the way it done in the Perceptron and in Adaline, requires long time and many learning steps. For this reason to solve the XOR problem it was needed to wait for faster and more adapt machines. Although learning process can be seen as an easy task in single layers perceptron, when introducing hidden layers thing start to be complicated. Rumelhart, Hinton and Williams[44] proposed a method to **backpropagate** the error across a the neurons in a network, allowing, so, to accelerate the learning process of some networks. This brought a final breakthrough to the XOR problems and multi-layers networks training, giving a concrete basis for the growth of Deep Learning field.

2.1.2 GPU and Tensorflow

Deep feed-forward networks usually take long time and a big amount of resource during the training. The backpropagation of the error and weights update are operations usually reserved to a Graphical Processor Units (GPU). GPU were invented by NVIDIA in 1999 and used to process the visual data. In 2007 NVIDIA released CUDA, a programming platform that allows parallel computing processes. 2 years later Raina [41] showed that machine learning tasks could have benefited of GPU usage processing tasks becoming 70 time faster then using a dual-core CPU. Tensorflow[1] is a python framework for Deep Learning written in C++ and CUDA. It was released by Google Brain Team in 2017, based on DistBelief which project was started in 2011. Tensorflow computations are represented in flow graphs and the operations declared can be easily specified to be loaded on GPU or reserved to CPU.

2.2 The State of the Art in Audio Domain

2.2.1 Wavenet

WaveNet[50] is a project conducted by Google Deepmind, first published in 2016. Wavenet consists of a convolution feed forward neural network using dilation layers to extrapolate inter time dependencies within one sound. The model can be used to generate music and speech. Global and local condition can be applied to specify for example to specify the gender of the speaker or the instrument to simulate and the phonemes or the pitch of the note that have to be synthesised. In the absence of local conditioning the network will generate something sounding similar respect to what it has been training on but not necessarily make sense to a human listener[2].

[2]Some audio example can be heard at `https://deepmind.com/blog/wavenet-generative-model-raw-audio/`

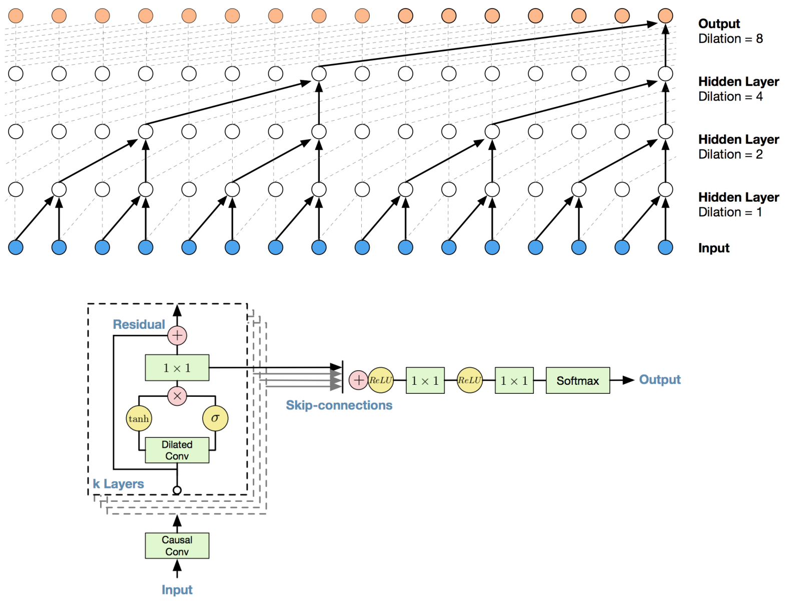


Figure 2.4: Convolution layer affected by dilation rate, Bottom: gate unit at the end Images taken from [50]

Similarly to two previous projects, PixelRNN[38] and PixelCNN[36], the algorithm has the particularity to predict one sample at a time. Given a set of samples form an audio junk, each of these samples is quantified in 128 channels. By doing so, each sample is linked to a distribution of possible values calculated by the neural network. The neural network calculates the probability distribution of the next sample and compares it to the one-hot vector of the original following sample. The internal layers have an interesting structure. To increase the length of the receptive field (i.e. the number of neurons from the first layer which are seen from one neuron in the output layer) by maintaining filter of short length and not increasing excessively the amount of layers, a dilation factor has been introduced to let the filter between two layers to pick non sequential samples (as shown in figure 2.4). This strategy allows the receptive field to grow exponentially respect to the depth of the network. The dilated convolution layers are collected in a stack where each layer receive as input the residual output of the previous one according to the activation gate shown in figure 2.4 summed to its input. The output of each layer is summed together and passed to a softmax function to calculate the probability distribution of the next sample. Unfortunately its original complexity didn't allow the algorithm to be used in real world application.

Parallel Wavenet[39], published in 2017, allows the algorithm to be implemented in real time application as it is currently implemented in Google Assistant[3] in English and Japanese version. Parallel Wavenet introduces the combination of three different losses. The first one is the Probability Density Distillation which is calculated as the Kullback-Liebler divergence between the output of a "*Teacher*" Wavenet network already trained and a "*Student*" Wavenet network which has to match the output of the distribution provided by the first one. The two networks are linked in cascade where the "Student" network plays the rule of Generator and the "Master" plays the rule of discriminator. The other two losses were introduced to improve the quality of the output: a power loss to ensure that the wide output of the model has similar frequency bands to the ones of the target sound, and a Perceptual loss to classify correctly the phones.

The layers used for the current project have been modelled similarly to the dilation layers used in WaveNet. Instead of one causal dilated layer, here is introduced a parallel version of it, each of the two parallel dilation layer is affected by different dilation rate in order to extrapolate features that are differently distributed along the signal.

2.2.2 NSynth

NSynth[9] is a neural synthesizer developed by Magenta[4]. Magenta is a research group originally started by some engineers and researchers from Google Brain Team whose goals is to provides Machine Learning tools for art and music purposes. NSynth consists of an autoencoder whose components are similar to what used in Wavenet network. In the paper was shown that Wavenet needs persistent external conditioning to generate stable outputs. NSynth aimed at introducing a model which is free of this need. To avoid the external conditioning the model decode for each set of 512 samples (corresponding to 32 ms of the original audio) of the audio chunk into an embedding space of 16 dimensions ($\times$32 compression) whose average is used as conditioning for a WaveNet decoder network whose input is the same original audio as shown in figure 2.5.

The network has been trained over a dataset of 300k notes 4 seconds long sampled at 16KHz from 1k musical instruments. The project provided a web interface to create new sounds by combining the properties of the embedding space learnt by the model over the dataset.

The current thesis is similarly structured to NSynth. Indeed, the two projects share the same goal, which is to create new sounds by sampling the latent space where are stored the latent representation of the training audios. Some differences have to be remarked: the thesis proposes an autoencoder that uses variational

[3]https://assistant.google.com/

[4]https://magenta.tensorflow.org/

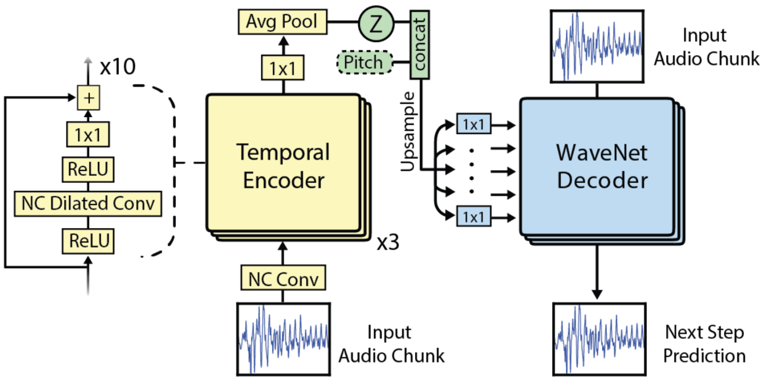


Figure 2.5: Structure of the WaveNet-style autoencoder used in NSynth. Image taken from [9]

inference and classification loss in the bottleneck to represent one entire sound in one 20 dimensional point. NSynth uses a 16 point-dimension for every 32ms of each audio. This choice allows NSynth to have a temporal representation in the bottleneck. This achievement can be easily reach by using raw autoencoders (see section 4.1.1), while the goal of this thesis was to build a latent representation not depending on temporal patterns. A second difference is that the NSynth uses a WaveNet decoder which regenerate the output sample by sample by using the quantized distribution for each sampel to be predicted, while the final layer of the model hereby proposed is composed of a convolutional layer allowing a simpler training.

2.2.3 Generative Adversarial Networks

Ian Goodfellow introduced in 2014 a new framework of generative models called Generative Adversarial Networks (GANs)[11]. The framework presets two networks which are train at the same time as shwon in figure 2.6. The first network is a generative network G while the second one is a discriminative network D. The G net tries is trained to reconstruct its input while the second network tries to discriminate the generated output from the real training sample. Usually both the networks consists of convolution layers. The output of the generated sample is usually trained respect to mean square error between its output and the training sample. The D net is usually implemented with a cross-entropy function to measure the difference between the probability distribution of the predicted classes and the ground truth of the sample (i.e. it is generated or it belongs to the training set of samples). The two networks are so in a competition way, where the generative network tries to learn a good representation of the image to fool the other one.

This framework has already been widely experimented in different fields and

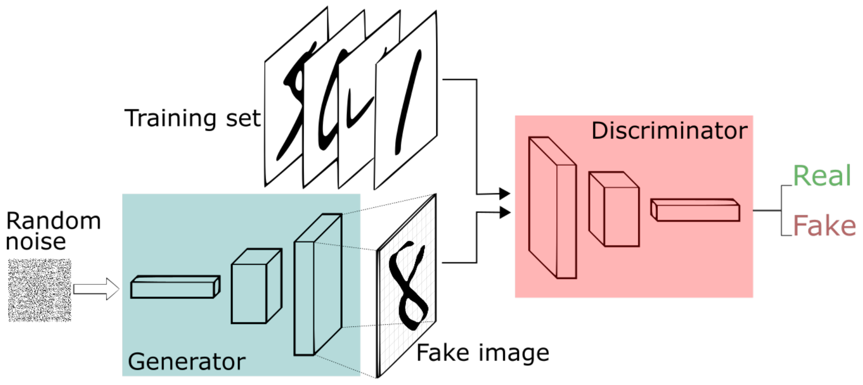


Figure 2.6: Representation of GANs architecture applied to images. Image taken from `https://deeplearning4j.org/img/GANs.png`

it brought interesting results for different tasks. By conditioning it is possible to use GANs to create images from some specified features such as labels, or edges [16]. CycleGan[59] train GANs to translate images between two different domain by using unpaired data. This allow for example to translate pictures in painting kind of styles, changing attributes from one classes to another one (like changing the season of a pictures, some objects appearing in the pictures with other objects). Changing the objects comparing in a picture is a challenging and interesting goal, for example GeneGANs[58] uses unpair data to add glasses, facial expressions and change the gender, hair style, skin color in portrait pictures.

GANs implementations to audio domain is also a wide area where researchers have already started to investigate in[57]. One example is the WaveGan[8] which use this framework for raw audio synthesis. A second example is using GANs to increase the resolution of the audio signals augmenting its sample rate[20]. Finally, Facebook AI Research group published[33] a project about music style transfer with an end-to-end approach by means of GANs structured through WaveNet-based encoders and decoders for conditioning the network.

The current project proposes in the conclusions an implementation of GANs to be used as fine tuning after the main training. The goals is to reduce the noise affecting the decoding of points between two latent representation of training audios.

2.2.4 Variational Autoencoder

Variational Autoencoder[19][18] (see section 3.3 for the formal introduction) is a method which infer to the latent representation values to approximate one probability distribution. This force the samples to be represented in a continuous of value of the latent space distributed according to some prerogative. Some studies[37] show that is also possible to infer the latent space with discrete representation by nearest neighbour calculation before fitting the sampled random variable representing the sample into the decoder.

A combination of variational autoencoder and gans applied to image domain was already proposed in [21][30]. The project used a combination of the two methods using portrait images reconstruction and changing of visual attributes mapped from the latent representations.

The autoencoders used in this thesis are inferred with a normal distribution at their bottleneck. An implementation of quantification of the latent space is let for a future work.

2.3 Related Projects Conducted by Students at Aalborg University Copenhagen

Researches in Deep Learning at Aalborg University Copenhagen campus, had been of interests during the last few years by many students. In the current and past years different project involving Deep Learning and sound contexts were done by students at Aalborg University Copenhagen. During the spring semester 2017, Deep Learning subjects were token as thesis subjects from mnay students. **Andrea Corcuera**[26] used Deep Learning techniques for impacts audio rendering in video. **Jose Luis Diez Antich**[3] brought a studies of audio events classification in a end to end approach.

Charles Brazier[5] studied pitch shifting by means of autoencoders. **Mattia Paterna**[40] performed spectrum features timbre transformation of piano notes through autoencoders.

In the current semester when this master thesis has been conducted, other two students, Aleix Claramount and Tomas Gajarsky from Sound and Music Computing MSc were involved in topics concerning in Deep Learning for sound generation and representation. **Aleix Claramount** carried a study focused on WaveNet implementing global and local conditioning to control music instruments generation. **Tomas Gajarsky** conducted different studies regarding Generative Adversarial Network, to generate accurate sounds from white noise. Previous academical project in machine learning I have been working on included: Long-short term memory networks for musical style detection, parametric t-SNE for browsing songs catalogue according to genre and moods features[23] and a study on vibrato features on violin expressive performance[24].

Similarly to some projects already done, the thesis project presented hereby is given by autoencoders. It propose a end-to-end approach using as input and output raw audio. However, multidimensional points from the latent space to be decoded into audio are used.

Chapter 3

Machine Learning and Probability Representation

3.1 Neural Networks

The current project used some instances of multi-layer Artificial Neural Network to develop the machine learning methodologies needed for the proposed study. Be $\{(\vec{x}_i, \vec{y}_i)\}_{i \in \{1,2,...N\}}$ a dataset of N paired vectors $\vec{x}_i \in \mathcal{X} \subset R^q$ and $\vec{y}_i \in \mathcal{Y} \subset R^p$. An Artificial Neural Network (ANN) is a computational model f_{net} which, given an input $\vec{x}_i$, maps an output $\vec{y}_i$ such that $f_{net}(\vec{x}_i) \simeq \vec{y}_i$. The input and the output are referred as input and output layers. ANNs are usually composed at least of one internal layer, that is called a hidden layer, which is composed by a certain amount of units, called neurons, each of them computing a linear combination of the outputs of the previous layer with a set of weights. The output of each neuron is then computed through and activation function (linear, relu, tanh, sigmoid, etc...). The output of the j_{th} layer, consisting of n neurons, is calculated as follows: $\vec{h_j} = g(\sum(\vec{h_{j-1}} W_j + \vec{b_j}))$, where W_j is a weight matrix of shape $\|\vec{h_{j-1}}\| \times n$, $\vec{b_j}$ is a vector of dimension n and g is the activation function. During the training process the weights $W_{j=1:numberoflayers}$ are adjusted to minimize a loss function that indicates the error between the actual output of the model $f_{net}(\vec{x}_i)$ and the real target $\vec{y}_i$. During the training process, the weights are continuously updated by feeding repeatedly batches from the set X into the network f_{net}. Once the training ends, the model, whereas the training reaches successful results, is ready to make prediction over new pair samples excluded from the samples used for the training. The goodness of the network is then evaluated respect to how well it is able to predicts correct values from the samples not seen by the network during the training.

According to the type of the data needed to be modelled, there exist different structures of neurons and layers. These allow to deal, for instance, with time dependencies (Recurrent Neural Network and Long Short-Term Memory, LSTM[14])

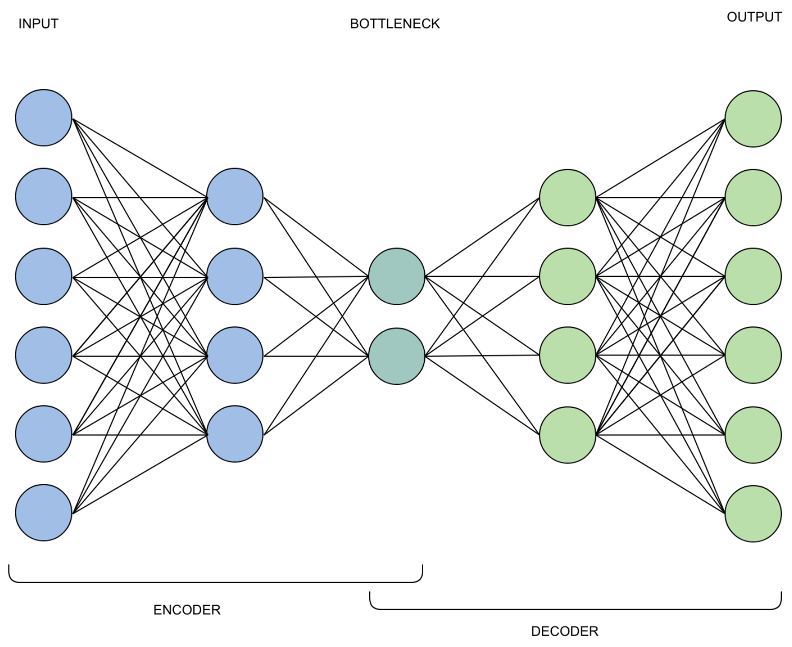


Figure 3.1: Representation of a 5-layers autoencoder with 6 neurons as input and output and 2 neurons in the bottleneck.

or with images (Convolutional Neural Network, CNN). LSTM is often used in natural language processing[53], while CNN are used mainly in image domain for classification and attribute localisation. Caption generation[55] is an interesting cross-field that uses both techniques to generate text description of features extracted from images. LSTM is a model provided with neurons capable to preserve information from sequence of samples flowing through the network and using it to influence the following samples. Its neurons are composed of multiple gates which control the information flowing throughout the model. CNN is a network whose weights consist in filters and whose layers are obtained by convolving the filters with the previous layer. The usage of convolution layers are useful in image domain and the filters are taught to extract multiple features and attributes from images (such as edges, lines, shapes, objects, etc...).

3.2 Autoencoder

The type of Neural Networks used in this thesis are called Autoencoders. Autoencoders are DNN whose goal during is to reconstruct the object given in input after a dimensionality reduction computed inside the network whose structure is represented in figure 3.1. An autoencoder is made of two components called respectively encoder and decoder. The encoder is structured by a sequence of layers which sequentially decrease the dimension of their outputs. By opposite, the decoder is a stack of layers whose dimension increase until reach the same dimension of the input layer. The architecture of encoder and decoder is usually symmetric. The bottleneck of this architecture corresponds to the layer at the end of the encoder and its dimension is the smallest among the layers of the network. According to the definition of DNN as a hierarchical feature extractor, the bottle neck represent the higher abstraction of the object to be reconstructed.

An autoencoder can be mathematically represented as a concatenation of two function θ and ϕ:

- $\theta : \mathcal{X} \rightarrow \mathcal{Z}, \phi : \mathcal{Z} \rightarrow \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^q$ and $\mathcal{Z} \subset \mathbb{R}^s$.
- such that $\theta, \phi = argmin_{\theta,\phi}\{L(X, (\theta \circ \phi)(\mathcal{X}))\}$ where L is a loss function $\mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ that calculate the dissimilarities between its arguments.

$\vec{x}_i \in \mathcal{X}$ is one original sample, $(\theta \circ \phi)(\vec{x}_i) = \vec{y}_i \in \mathcal{Y}$ is the reconstruction of the datapoint $\vec{x}_i$, while $\vec{z}_i = \theta(\vec{x}_i) \in \mathcal{Z}$ is the latent - or coded - representation of $\vec{x}_i$. θ and ϕ are respectively the encoder and the decoder. $\mathcal{Z}$ is the bottleneck of the autoencoder and $\phi(\vec{x}_i \in \mathcal{X})$ is the latent representation of the data $\vec{x}_i$ computed by the encoder.

3.3 Variational Autoencoder

"Generative Modelling" is used in machine learning to generate data whose probability representation is learned to be similar to the dataset used in the training. On the contrary, the goal in discriminative models[4] is to assign for each feature vector $\vec{x}_i \in \mathcal{X} \subset \mathbb{R}^q$, one or multiple labels among q classes $\vec{c}_i \in \mathcal{C} \subset \{0,1\}^q$ to teach a model to classify a new vector $\vec{x}^*$, with same dimensions of the elements in the set $\mathcal{X}$, with the label $\vec{c}^*$. This is achieved by computing $P(\vec{c}^*|\vec{x}^*, \vec{x}, \vec{c})$ after learning the likelihood $P(\vec{x}|\vec{c})$ from the reference dataset $\{(\vec{x}_i, \vec{c}_i)\}_{i\in\{0,1,...N\}}$.

On the other hand, generative models try to learn the join probability distribution[48][35] that describes the dataset and then to generate new data by sampling the obtained distribution.

The following explanation is adapted from [7] and [42]. In variational autoencoder, given a datapoint $\vec{x} \in \mathcal{X}$, the idea is to sample a latent random variable $\vec{z} \in \mathcal{Z} \subset \mathbb{R}^{p<q}$ from the the probability distribution $P(z)$ defined over $\mathcal{Z}$, such that a decode function $f : \mathcal{Z} \times \Theta \rightarrow \mathcal{X}$ for some space Θ, returns an object $f(z, \theta \in \Theta)$ similar to x. The joint probability of the model is $P(x,z) = P(x|z)P(z)$. z random variable is drawn by the prior $\sim P(z)$ while $x \sim P(x|z)$ is the likelihood of the framework. The goal is so to learn the posterior mapping $P(z|x) = \frac{P(x|z)P(z)}{P(x)}$.

By replacing the function $f(z,\theta)$ with the probability distribution $P(x|z,\theta)$ and

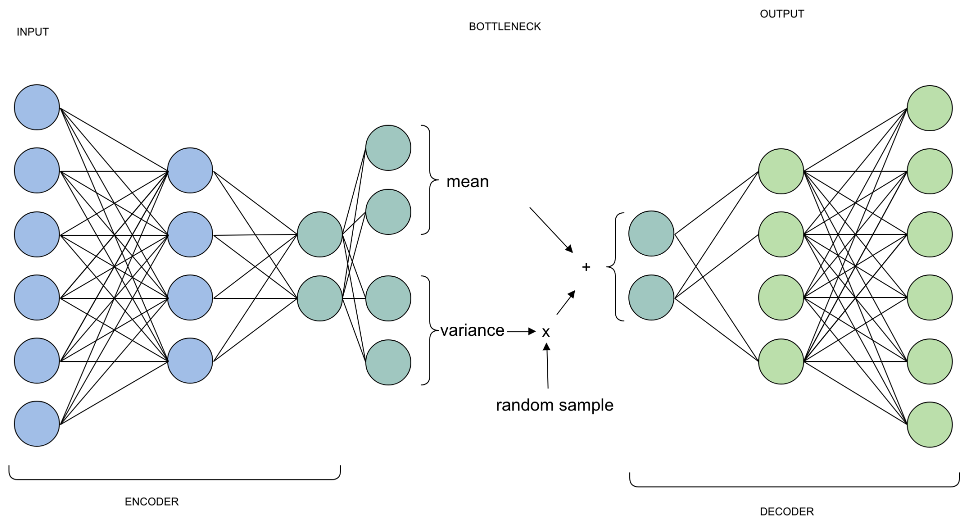


Figure 3.2: Example of Variational Autoencoder. The dimension of the mean and the variance of the latent representation of the bottleneck coincides, in this example, with the dimension of the last layer of the encoder. The variance neurons are multiplied by a random sample and summed with the mean before fired inw the decoder.

by applying the law of total probability, we obtain the following:

$$P(x) = \int P(x|z,\theta)P(z)dz$$

The choice of the latent variable z is often a normal distribution $\mathcal{N}(0, I)$ such that $P(x|z,\theta) = \mathcal{N}(x|f(z,\theta), \sigma^2 I)$. To maximize $P(x)$, there is needed an exponential time to solve the above equation since it needs to evaluate over all the configurations of the latent variable. To solve this problem it is possible to infer the latent space with respect to a family distribution $Q_\lambda(z|x)$, where λ is the variational parameter $\lambda_{x_i} = (\mu_{x_i}, \sigma^2_{x_i})$. Thus, the approximated posterior is $Q_\theta(z|x,\lambda)$ which is the probability function that describes the encoder: $x \mapsto \lambda$. $P(x|z)$ corresponds, instead, to the generative network which consists in a decoder: $\lambda \mapsto P_\phi(x|z)$. The difference between the approximation and the real posterior is obtained by means of Kullback-Leibler divergence $KL(Q_\lambda(z|x)\|P(z|x))$:

$$KL(Q_\lambda(z|x)\|P(z|x)) = \mathbb{E}[logQ_\lambda(z|x) - logP(z|x)] \tag{3.1}$$

$$KL(Q_\lambda(z|x)\|P(z|x)) = \mathbb{E}[logQ_\lambda(z|x) - logP(x|z) - logP(z)] + logP(x) \tag{3.2}$$

$$logP(x) - KL(Q_\lambda(z|x)\|P(z|x)) = \mathbb{E}[logP(x|z)] - KL(Q_\lambda(z|x)\|logP(z)) \tag{3.3}$$

To perform stochastic gradient descent on the right hand side of the previous equation, we can chose $Q_\lambda(z|x) = \mathcal{N}(z|\mu(x,\theta), \Sigma(x,\theta))$. By doing so, the

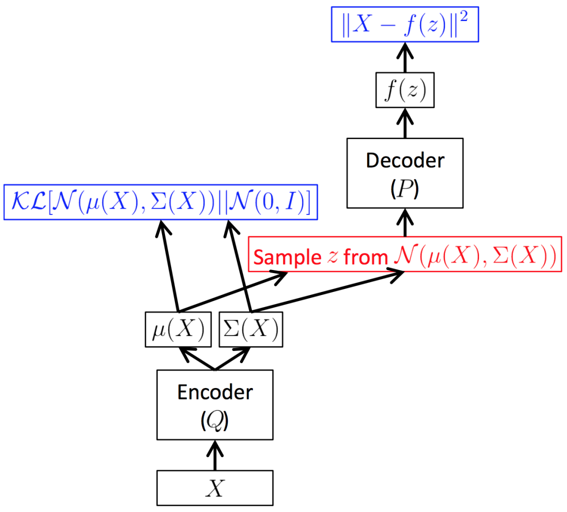

$\|X - f(z)\|^2$
$f(z)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X))\|\mathcal{N}(0, I)]$
Sample z from $\mathcal{N}(\mu(X), \Sigma(X))$
$\mu(X)$
$\Sigma(X)$
Encoder (Q)
X

Figure 3.3: The graph shows the loss functions of an autoencoder model inferred with Normal distribution approximation at the bottleneck. The image is taken from [7]

Kullback-Liebler divergence is calculated between two multivariate Gaussian distribution. By following [7], the Kullback-Liebler divergence between two Gaussian distribution is equal to

$$KL(\mathcal{N}(\mu_0, \Sigma_0)\|\mathcal{N}(\mu_1, \Sigma_1)) = 0.5(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) - k + log\frac{|\Sigma_1|}{|\Sigma_0|}) \tag{3.4}$$

$$KL(\mathcal{N}(0, I)\|\mathcal{N}(\mu(x), \Sigma(x))) = 0.5(tr(\Sigma(x)) + \mu(x)^T\mu(x) - k + log|\Sigma(x)|) \tag{3.5}$$

$\mathbb{E}[logP(x|z)]$ is given by the approximation of $P(x|z)$ for one sample of z. z is so sampled from $\mathcal{N}(\mu(x), \Sigma(x))$ as $z = \mu(x) + \Sigma^{1/2}(x) * \epsilon$, where ϵ is sampled from $\mathcal{N}(0, I)$ as shown in figure 3.2. By considering $P(x|z) \sim \mathcal{N}(f(z), \sigma^2 I)$, the log-likelihood is given as $logP(x|z) = C - 0.5\|x - f(z)\|^2/\sigma^2$ where C is a constant and σ can be treated as a weight factor. As it is summarised in figure 3.3, the objective function of a variational autoencoder is:

$$\|\phi(\psi(x)) - x\|^2 + \|\psi(x)\|_0 \tag{3.6}$$

where ψ and ϕ are respectively the encoder and decoder networks.

The loss can be interpreted as the sum of a regularizing term (the Kullback_Liebler divergence) with the generation loss (the expected likelihood). $Q_\theta(z|x, \lambda)$ is inferred by the encoder while $P_\phi(x|z)$ is parametrized with a generator network - the decoder.

The last layer of the encoder consists of two parallel dense layers whose job is to calculate for each input datapoint, the variance and the mean of its correspondent point distribution in the latent space.

Python code: Q(z|x)

```
current_layer = encoder(input)
z_mean = tf.layers.dense(current_layer, n_hidden,
     activation = None))
z_var = 0.5 * tf.layers.dense(current_layer, n_hidden,
     activation = None)
epsilon = tf.random_normal([tf.shape(current_layer)[0],
     n_hidden], 0.0, 1.0)
z  = z_mean + tf.multiply(epsilon, tf.exp(z_var))
output_layer = decoder(z)
```

Python code: Loss calculation

```
generation_loss = tf.reduce_sum(tf.squared_difference(
   output_layer, input), 1)
latent_loss = tf.reduce_sum(tf.log(sigma0)-z_var+
   0.5*tf.square(tf.exp(z_var)/sigma0)+
   0.5*tf.square((z_mean-mean0)/sigma0),1)
loss = generation_loss + latent_loss
```

Chapter 4

Methods

When dealing with complex systems characterised by many parameters, one particularly hard task consists in understanding the contribution of their components and how the changing of their parameters affects the entire model. The experiments described in the current and following chapters, were run on a local server from which one GPU slot provided with NVIDIA GeForce GTX Titan X 12GB was used. The models are completely written and run using Tensorflow[1], a Deep Learning library for Python.

4.1 Autoencoder structure

Three autoencoders with different architectures were analysed to study the basic representation that autoencoders store in their bottleneck to describe the audio input:

- convolution based autoencoder,
- dilation based autoencoder,
- hybrid autoencoder.

Here follows a description of the layers composing the autoencoders. An analysis of the latent representation is then given. The architecture details of the first two models proposed are shown in table 4.1.

The convolution based autoencoder consists of a stack of 12 convolution layers. The first 6 layers compose the encoder. The amount of filters for each layer is equal or larger than the previous except for the last layer. This provides a tensor with 1024 channels in the last layer before the bottleneck. The kernel size is equal to 5 for the first layer while is fixed to 4 for the rest of them. Each of the first 5 layers has a stride factor equal to 2. By doing so, the length of the tensor at the bottleneck is decreased by a factor of $2^5 = 32$. wThe bottleneck consists of a convolution layer

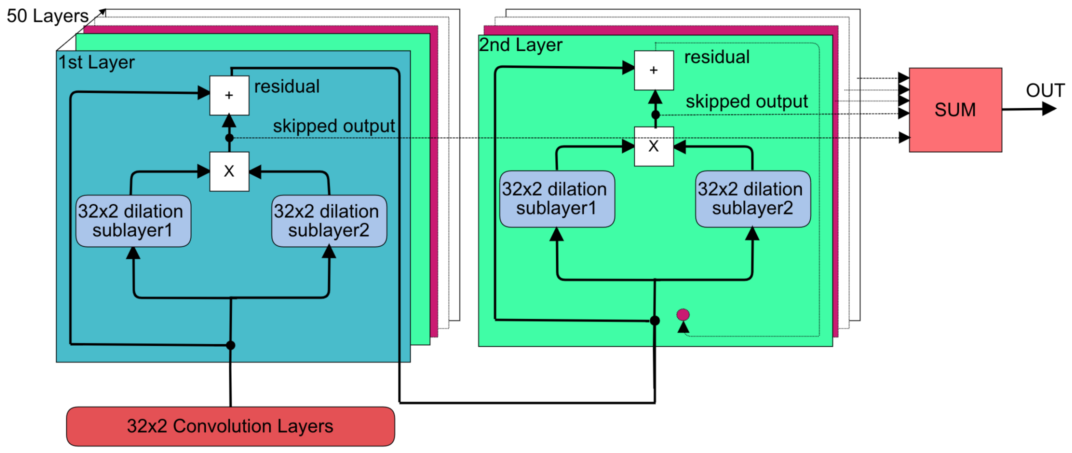


Figure 4.1: Representation of one dilation block used in the dilated and hybrid model.

1×1 used to collapse the 1024 channels into a single channel. The architecture of the decoder is exactly symmetrical. The activation function used in all the layers excluded the bottleneck layer, is an hyperbolic tangent. The bottleneck layer has been tried with both hyperbolic tangent and a linear activation function. These parameters have been decided after some trials and similarly to the convolution autoencoder model used in [9].

The second model, has an architecture inspired by the skip outputs and dilation layers used in WaveNet (see section 2.2.1). After a first convolution layer (16 filters 32 samples long, with stride 1), a block of dilation layers is applied followed by 5 layers with 1×1 filters and stride equals to 2. The rest of the dilation based autoencoder is composed by a decoder symmetrically set. Figure 4.1 shows what happens inside a dilation block. The dilation block consists of 50 residual layers [13] each of them composed of 2 parallel sub-layers $\text{sub\_layer}^{1,2}_{i=1:50}$ of 32×2 filters affected by a dilation rate. The dilation rate applied to the i_{th} layer is $2^{mod(i-1,m)}$, where m is respectively equal to 10 and to 5 for the two parallel sub-layers. The receptive field seen by one output neuron from the last, 50th layer of the dilation block is equal to:

$$recf = \sum_{i=1}^{N=50} 2^{mod(i-1,m)} + 1 = \frac{N}{m}(2^m - 1) + 1$$

We so obtain $recf\ |_{m=10} = 5119$, $recf\ |_{m=5} = 319$. The input of the following i_{th} layer in the dilated block is given by

$$\text{input}_i = \text{input}_{i-1} + \text{sub\_layer}^1_{i-1} \cdot \text{sub\_layer}^2_{i-1}$$

The output of the entire block is given by the average of all the 50 multiplications of sublayer outputs from each layer:

$$\text{output\_block} = \frac{1}{50} \sum_{i=1}^{N=50} \text{skip\_output}_i,$$

with $\text{skip\_output}_i = \text{sub\_layer}^1_{i-1} \cdot \text{sub\_layer}^2_{i-1}$. This procedure allows the network to extract information regarding temporal patterns across the input audio.

The hybrid model is a combination of the previous two models, providing a concatenation of one stack of convolution layers with the output of the dilation block for the encoder, vice versa for the decoder. The architecture of the layers in the hybrid model is similarly structured like the previous two models and is shown in table 4.2.

All the layers in the models were passed through an added batch normalisation layer. Drops out layers were also considered to be used, but because of the conflicts between the two techniques[22], the choice felt on using only batch normalisation layers.

CONVOLUTION BASED AUTOENCODER:

#	layer type:	filters:	kernel:	strides
1	conv1d	128	5	2
2	conv1d	128	4	2
3	conv1d	256	4	2
4	conv1d	512	4	2
5	conv1d	1024	4	2
6	conv1d	1	1	1
7	conv1d_transpose	1024	4	2
8	conv1d_transpose	512	4	2
9	conv1d_transpose	256	4	2
10	conv1d_transpose	128	4	2
11	conv1d_transpose	128	5	2
12	conv1d_transpose	1	1	1

DILATION BASED AUTOENCODER:

#	layer type:	filters:	kernel:	strides
1	conv1d	16	32	1
2	50 x dilation_block	32	2	1
3:7	5 x conv1d	1	2	2
8	conv1d	1	1	1
9	dense_a dense_b size: 20	-	-	-
10	add	-	-	-
11	conv1d_transpose	1	1	1
12:17	5 x conv1d_transpose	1	2	2
18	conv1d_transpose	32	1	1
19	50 x dilation_block	32	2	1
20	conv1d_transpose	16	32	1
21	conv1d_transpose	1	1	1

Table 4.1: Left: Architecture of the baseline model made of 12 convolution layers. Right: Architecture of the dilation based model made of 21 layers

HYBRID AUTOENCODER:

#	layer type:	filters:	kernel:	strides
1	conv1d_1	16	32	1
2	50 x dilation_stack	1	-	1
3	conv1d_2	128	4	2
4	conv1d_3	128	4	2
5	conv1d_4	256	4	2
6	conv1d_5	256	4	2
7	onv1d_6	512	4	2
8	conv1d_7	1	1	1
9	conv1d_transpose_8	512	4	2
10	conv1d_transpose_9	256	4	2
11	conv1d_transpose_10	256	4	2
12	conv1d_transpose_11	128	4	2
13	conv1d_transpose_12	128	4	2
14	conv1d_transpose_13	1	1	1

Table 4.2: Architecture of the model combining the convolution and the dilation based autoencoder.

4.1.1 How is the latent information represented in autoencoder?

To study the three models introduced above, some audio datasets were programmatically built. The datasets used were composed of modulated sine waves half a second long with a sample rate of 16Khz sharing all the same constant amplitude. The datasets included a set of sine waves with dynamic and constant frequency parameters:

- sine waves with time-constant frequency $\in [80, 1200]$Hz
- sinewaves with linearly-changing frequency, from an initial frequency $\in [80, 1200]$Hz and final frequency twice its initial value.
- frequency modulated sine waves with time-constant carrier frequency $\in [100, 800]$Hz and linearly-changing frequency and amplitude of the modulator from an initial to a final value in the respective ranges $[3, 100]$Hz for the frequency and $[1, 200]$Hz for the amplitude.

Some pilot experiments were conducted by considering also polyphonic sounds computed through additive synthesis. However, they showed worse results for the reconstruction task compared to the datasets listed above.

The three models were trained with the frequency modulation dataset. The models were trained with 7000 frequency modulated sine waves by using Adam Optimizer and mean square error as loss function. The sine waves were collected in batches of 500 elements each reapeted for 70 iterations. Figure 4.3 shows some examples of the spectrogram obtained in the reconstruction task for the frequency modulation dataset after the training was completed. The models were then used to analyse how the latent information is composed, by feeding the models with samples from the other two datasets.

By considering the $\times 32$ compression factor applied to the input signal at the bottleneck, the length of the signals is reduced from 8192 samples to 265 samples in which half of a second of information is stored. To investigate to what extend the latent representation might still be a time sequence representation of the original signals, the latent representation is assumed to be a signal whose sample rate is reduced by the same compression factor affecting the length of the audios. By doing so, the correspondent sample rate in the latent domain is obtained and it is equal to 500Hz. From the comparison of the spectrograms of an original signal and its latent representation, it is seen that the representation at the bottleneck is still a signal that maintains the same frequency of its input. One important observation is that the scaled sample rate implies a new limit superior for aliasing because of the Nyquist–Shannon sampling theorem. Because of this, the signals that can be represented in the bottleneck without being affected of aliasing, are reduced to all those signals whose frequency is less than half of the latent sample rate. One interesting point is so to understand what happens to the latent representation in the neighbourhood of the maximum frequency presentable. In figure 4.2, we can see that the latent representation of the sine wave is not just a down-sampled version of it. In order to avoid the repercussions of aliasing, the autoencoder seems to learn a non symmetric waveform and to modulate in amplitude this representation as shown in figure 4.2.

4.2 Variational Autoencoder

So far, it has been shown that the bottleneck of the autoencoders learns a *temporal sequence* representation of the input signal. As seen in figure 4.3, this latent representation provides a good enough tool to reconstruct a simple signal. However, looking at editing this latent representation in order to manipulate the output of the autoencoder, this approach does not provide a suitable representation of the signal. Ideally, we would like to have a latent space where each of its point provides the latent representation of a different audio and perhaps collapsing similar signal in the same area of the latent space. To reach this goal, it is needed to free the latent projection from the temporal sequence encoding of the original object by forcing the encoder to follow some criteria. What it is here introduced is a model that can provide a easily tractable latent representation to generate, from each of its point, new objects stochastic consistent to the training dataset.

4.2.1 Experiments by using variational inferences

The structures of the encoder and decoders are similar to what presented in tables 4.1 and 4.2. Between the encoder and decoder there are introduced two parallel dense layers to store the mean and variance information of the audio sample. Then,

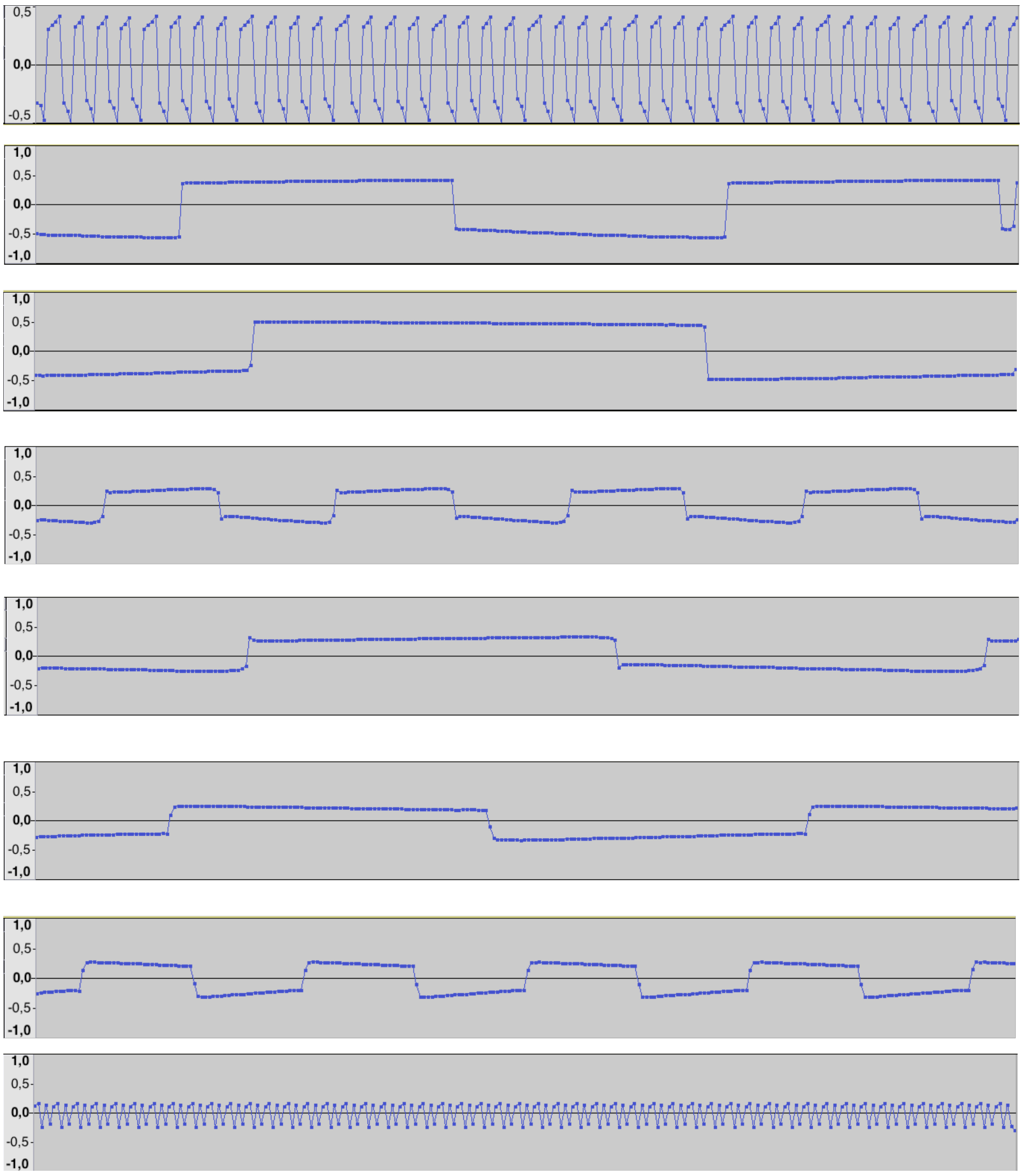


Figure 4.2: Latent representation affected by aliasing of sine waves with time-constant frequency at the bottleneck from the convolution based autoencoder trained through dynamic frequency modulators. From top to bottom: 80.0Hz, 248.0Hz, 253.6Hz, 992.8Hz, 998.4Hz, 1004.0Hz, 1009.6Hz, 1200.0Hz.

an additive layer to sample the latent sample z is provided followed by a sequence of two dense layers to increase the length of the layer till the dimension of the last layer in the encoder. The number of neurons of the two parallel dense layers determines the dimension of latent space:

Python code: Bottleneck (extended)

```
current_layer = tf.contrib.layers.flatten(decoder_layer)

mn = tf.layers.dense(current_layer, n_hidden)
sd = 0.5 * tf.layers.dense(current_layer, n_hidden)
epsilon = tf.random_normal(
      [tf.shape(current_layer)[0], n_hidden], 0.0, 1.0)

z  = mn + tf.multiply(epsilon, tf.exp(sd))

current_layer = z
current_layer = tf.layers.dense(current_layer, sample_length/32/2)
current_layer = tf.layers.dense(current_layer, sample_length/32)
current_layer=tf.reshape(current_layer,
      (-1,sample_length/32,1,1))
```

Some experiments have been conducted with simple signals to investigate the validity of the model proposed. The dataset used in this section is composed of 3 families of signals: sine waves 1 period long, saw-tooth of 2 periods, and Gaussian modulated sinusoids,each of them 512 samples long. The training set contained in total 2K signals, with values ranging from -1 to 1 summed with random white noise signal spacing in the range [-0.05,0.05]. The dataset has been fitted in the three models introduced in 4.1, augmented by the bottleneck to support the variational inference. The encoder reduces the length of the representation by a $\times 32$ factor, reducing so the length of the input to 16 samples. The dimension of the latent space was fixed to 2. As described in section 3.3, the loss function used is the reconstruction loss calculated as mean square error between input and reconstructed output, regularized by the variational loss at the bottleneck calculated as the Kullback-Liebler divergence between a Gaussian distribution and the values stored in the z latent sample. In the table 4.3 follow some graphic results of the models obtained. From the scatter plots shown in that table, we can see how a different combination of layers influences the organisation of the representation of the signal in the latent space. Although the lack of labels, the models organise the signals affected by random noise in clusters of signals from the same family. The convolution based variational autoencoder collects in the same neighbourhood the gaussian modelled sin waves and the sawtooth signals, while the sine waves are collected in a far small area. The variational autoencoder based on dilation blocks projects the signals in the latent space in non-overlapping regions. The hybrid autoencoder brings a latent space where the sawtooth signals are far separated from the other two signals' instances, however, the the gaussian modelled sine waves

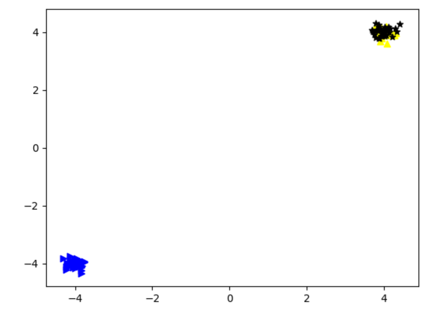
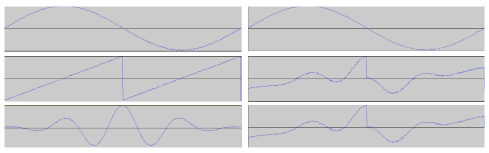
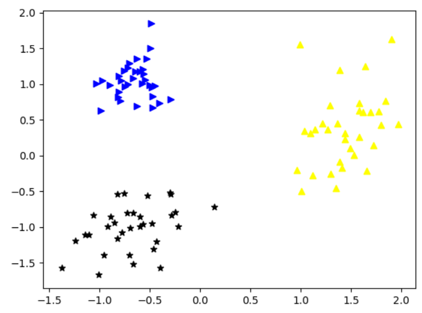
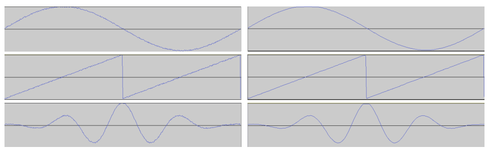
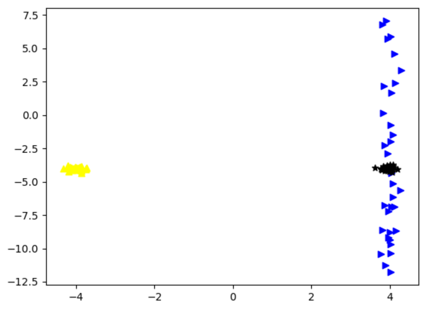
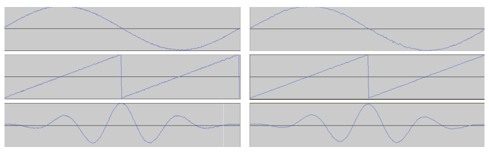
2D latent space*:	Input and output of the models:
	
	
	

Table 4.3: For each of the models introduced in section 4.1, the first column shows the distribution of the testing signals in the 2 dimensional latent space, the second column collects for each model three pairs of input and output, one from each family of signals used (sine waves, sawtooth, and gaussian modelled sinusoid). From top to bottom: convolution autoencoder, dilated autoencoder, hybrid autoencoder.
Legend for the scatter plots: blue ’>’ sine waves; black ’’ gaussian modelled sine waves; yellow ’ˆ’ sawtooths

overlap the the sine waves.

Once obtained a model with non-overlapping regions of samples referring to different signals, it comes easily to investigate what happens in the points not referring to any signal appearing in the dataset. Particularly, we can calculate the mean for each cluster and then pass its coordinates to the decoder to output the signal generated from that point. Moreover, it is possible to draw a path in the latent space, taking a set of coordinates by sampling a line starting from one point *a* in region *A*, to a different point *b* in region *B* as it is shown in figure 4.4. Figure 4.5 collects the sort outputs of 9 latent coordinates from the latent space corresponding to a path from the centre of the cluster of the latent projection of sine waves to the centre of points referring to the sawtooths. This allows to create morphing between the two objects that live in two separate regions of the latent space.

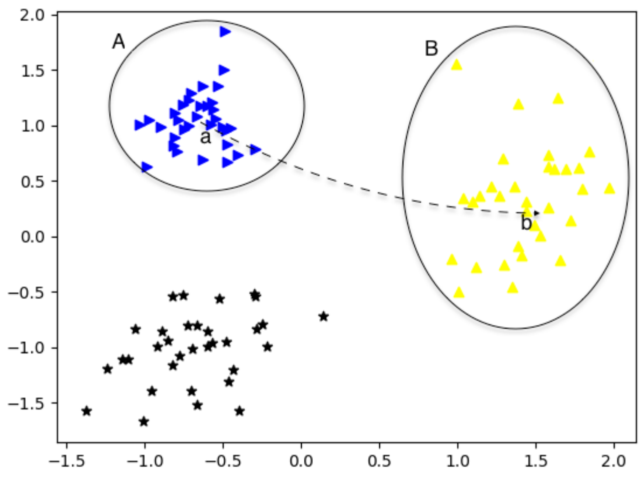


Figure 4.4: The figure shows an example of path drawn in the latent space. each point in a path connecting a point *a* to a second point *b*, refer to a particular output of the decoder network. By doing so it is possible to create morphing between two two regions *A* and *B* collecting the latent coordinates of different objects.

In figure 4.5 there is shown a morphing between a sinusoid and a sawtooth performed by the dialtion based model with latent space dimension set to 2. Furthermore, it is possible to see how the first output and the last output - referring respectively to the mean of the latent coordinates of sine waves and sawtooth - seem to be less affected by the noise respect to what was used in the dataset. This is because the point in the middle of a cluster is a compromise between all the points surrounding it. Since the points are all references of the same object affected by white noise, it makes sense that the point in the middle refers to a average between the signals projected in its neighbourhood, weighted by the distance between the centre and the points considered. This allows the representation of the signal, stored in the middle of a cluster of instances of the same signal affected by random noise, to be decoded as a clean signal. This phenomenon is known as the denoising property of variational autoencoders[15].

An example of the denoising property is shown in table 4.4. It shows the reconstructed middle point of the clusters by using the same dataset affected by the 20% of noise instead of the 5% used before.

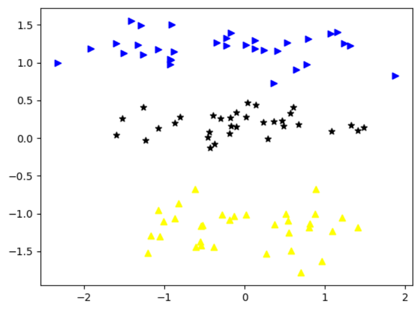
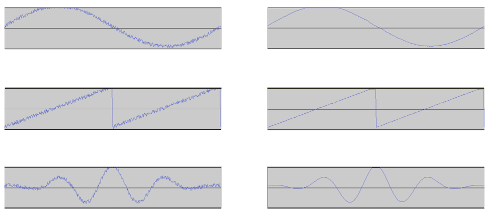
2D latent space*:	Training signals and outputs from the sampled latent space:
	

Table 4.4: Noise reduction application of variational autoencoder. The left image shows the scatter plot of the latent space of the dilation based autoencoder model trained on sine waves, sawtooth and gaussian modelled sine waves affected by white noise. On the right side there are shown in the first column three instances of the signals used during the training, while on the second column there are the outputs of the models for the avarage coordinate in the latent space of the correspondent signal. *Legend for the scatter plots: blue ’>’ sine waves; black ’*’ gaussian modelled sine waves; yellow ’ˆ sawtooths

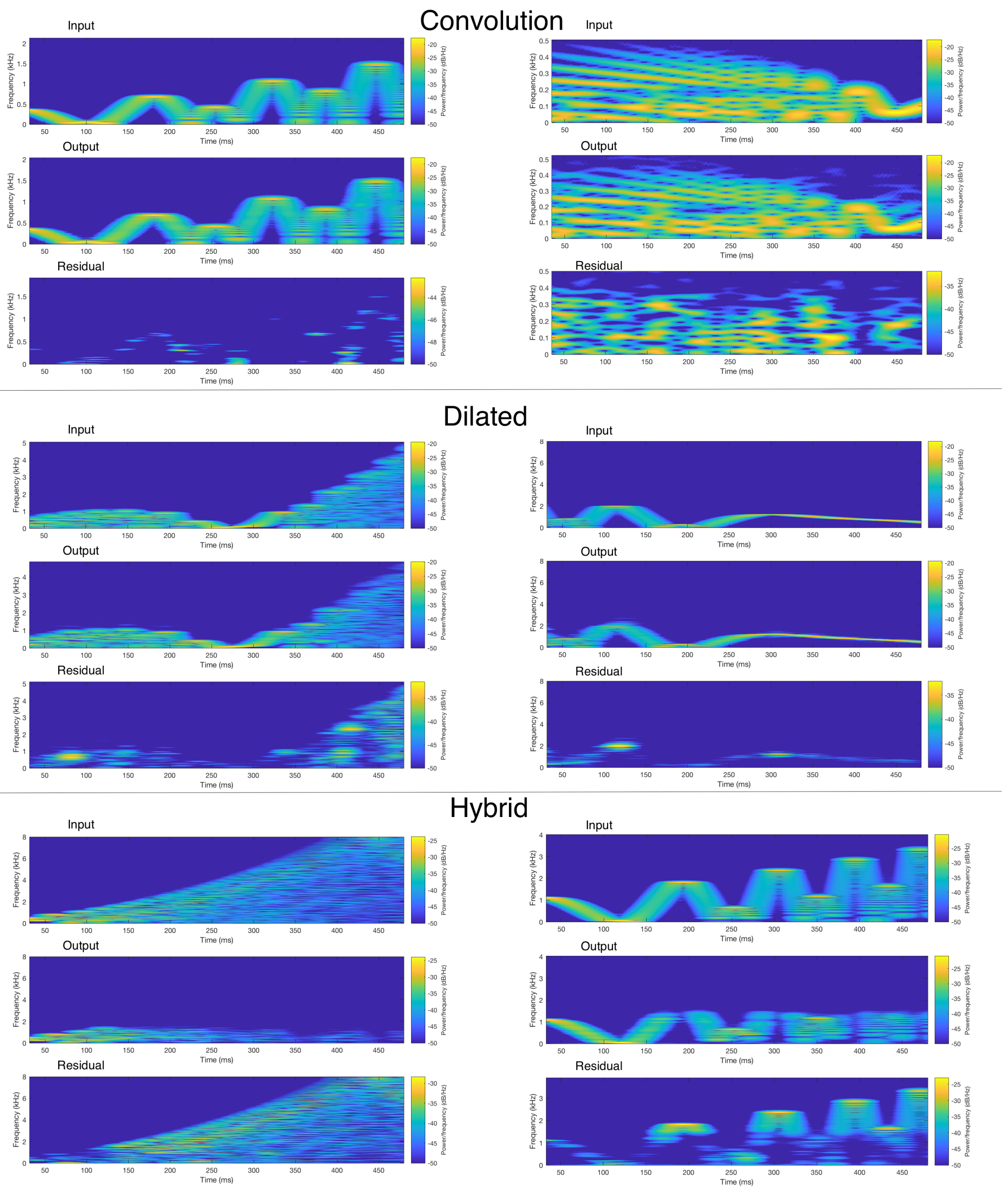


Figure 4.3: Set of spectrograms of input, reconstruction and residual for the convolution, dilated and hybrid models.

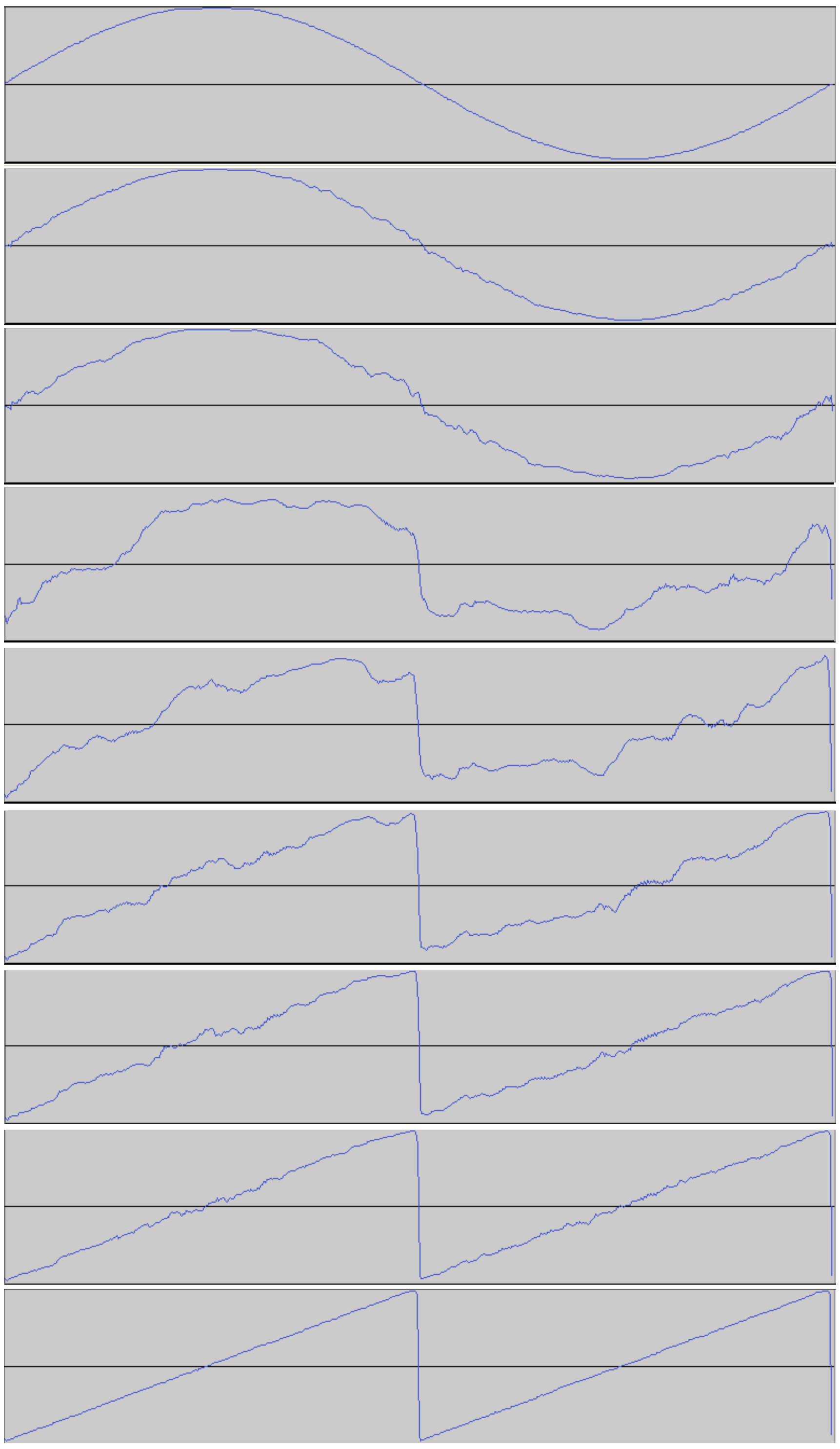


Figure 4.5: Morphing between a sine wave and a sawtooth. The morphing has been obtained by sampling the latent space moving from the middle point of the sinusoids cluster to the middle point of the sawtooth cluster.

Chapter 5

Results

5.1 Applying Variational Autoencoder to Audio Domain

The passage to audio domain requires some adjustment in the models seen so far. The dataset used in this part consisted of spoken digits and it is a subset of the database available at `https://github.com/Jakobovski/free-spoken-digit-dataset`. The digits {0,1,...,9} were recorded by one male speaker, repeating each digit 50 times. The audio files consist of a mono track audio with 8KHz of sample rate. All the audio files were zero-padded to be 4096 samples long, corresponding approximately to half a second of recording. To work with these files, the hidden space is set to 20 dimensions. In order to display it, two projection techniques onto a bi-dimensional space are used: a **Principal Component Analysis** (PCA) and a **t-distributed Stochastic Neighbor Embedding**[25] (t-SNE). On one hand, the PCA method projects the dataset over a space dictated by the n (in this case n is fixed to two) eigen-vectors with largest eigen-values from the covariance matrix of the dataset. On the other hand, t-SNE projects the dataset in a low dimensional space trying to maintain the pairwise distances similarity among the objects in the original high dimensional space, by calculating the distances as Gaussian probabilities distribution. The resulting projection is obtained by minimizing, through a gradient descent method, the Kullback-Liebler divergence between the original data distribution in the high dimensional space and its projection in the low dimensional space. The two methods provide two different information regarding the quality of the 20 dimensional latent distribution of the results obtained from model. PCA method provides the two largest directions across which the datapoints are spread, while the t-SNE method provides information about local distributions and neighbourhoods.

The figures in the table 5.1, show the complexity of spoken digits requires a bottle neck of higher dimension respect to what needed by the simple signals used in the previous chapter. It is easy to see how already in the first two columns audios of same digit are gathered significantly close to each other in the latent space. However, looking at the t-SNE projection in the last row of table 5.1, we can see that the increase of the dimensions is not enough to allow the model to collect the digits in non overlapping regions. This brings some problems wherever we would like to perform morphing operation between two digits.

Since the dataset provides labelled data, to solve this problem, a second loss function to perform classification on the bottleneck was introduced.

5.1.1 Adding a Classifier at the Bottleneck

The idea is to adapt the encoder to organize the datapoints in the bottleneck to be correctly classified by a new network. The new network consists of a 3 dense hidden layers-network, whose input is the sampled z obtained in the bottleneck and whose output fire as many neurons as the amount of classes considered (in this case 10). Each neuron in the output layer is associate to a different digit and a cross-entropy cost is calculated between the output of the classifier and the one-hot function of the target digit. During the training the model is adjusted with respect to the two gradients. It tries, so, to optimize both of them sequentially at each training iteration. The model adapt its weights to reach a Nash Equilibrium between the two cost functions.

Python code: Bottleneck Classifier

```
class_layer = tf.layers.dense(encoded_layer,10,activation=tf.nn.relu,
    trainable=True,kernel_initializer=tf.random_normal_initializer)
class_layer = tf.layers.dense(class_layer,10,activation=tf.nn.relu,
    trainable=True,kernel_initializer=tf.random_normal_initializer)
class_layer = tf.layers.dense(class_layer,10,activation=tf.nn.sigmoid,
    trainable=True,kernel_initializer=tf.random_normal_initializer)
class_output = tf.nn.softmax(class_layer)
```

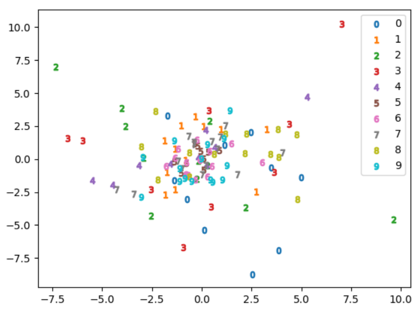
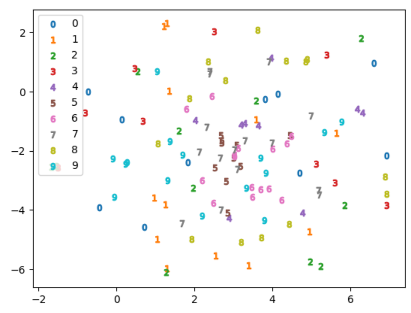
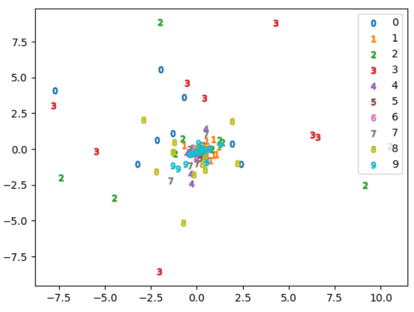
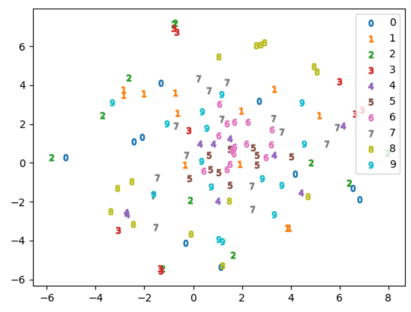
5.2 Evaluation of the latent organization

The table 5.3 shows the latent distribution of the variational autoencoder models based on the three autoencoders introduced in 4.1 by setting the latent space to 20 dimensions. The scatter plots are the PCA and t-SNE projections from the 20-dimensions latent space into a bi-dimensional space. The results are obtained from the testing set composed of 10 audio per digits, after a training of 1000 iteration. It is easy to see that the hybrid model gets worse clustering results than the dilation and convolution based models.

1-Nearest Neighbour (1-NN) is performed to evaluate the distribution of the

PCA projection: t-SNE projection:

Latent Space 10 dim:

Latent Space 32 dim:

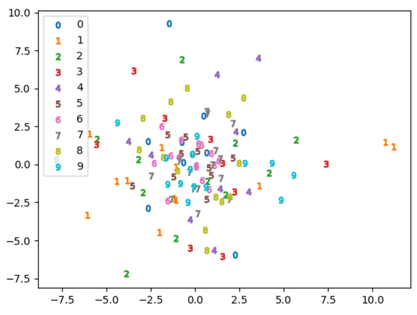
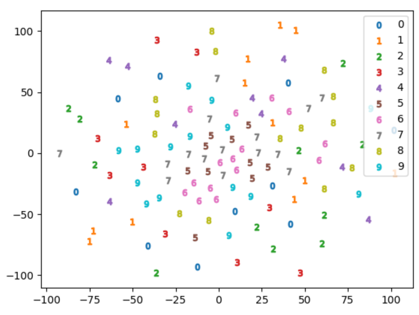



Table 5.1: The figures represent the latent space of a convolution based variational autoencoder fitted with the spoken-digit dataset. The first two rows are obtained by two different training by setting the dimension of the latent space to 10, while the model referred in the third row has a dimensional space fixed to 32. The digit symbols represent the actual digit to which the input audio refer.

projection of the instances in the latent space. 1-NN is a particular case of k-NN classification and regression machine learning technique that assigns to each point in the features space the most common class or the average value among

Model:	**1-NN Accuracy:**
Convolutional VAE	0.94
Dilated VAE	0.96
Hybrid VAE	0.09

Table 5.2: 1-NN accuracy results across the bottleneck.

its k nearest datapoint from the training set. For each test sample, the 1-NN his defined over all the datapoints in the latent space except for one and a digit class for the excluded datapoint is so predicted. The table 5.2 presents the results of this evaluation. We can see in table 5.3 the confirmation of what seen from the scatter plots: the hybrid model reaches poor results in organising together the datapoints in the latent space that belong to the same class, indeed only 9% of the latent points are close to a point belonging to the same class. Better results are achieved by the other two models. Convolution and dilation based models achieve respectively 94% and 96% of accuracy.

5.3 Evaluation of the audio consistency

A combination of **Dynamic Time Wrapping**[45] (DTW) and **Mel Frequency Cepstrum Coefficients**[29] (MFCC) algorithms is used to evaluate the consistency of the generated audio from the middle point of each cluster of latent points. Combination of DTW and MFCC is one of the most common strategy used in speech recognition[32, 34, 12, 31, 51]. DTW provides a distance measurement between two signals by considering similar patterns dilated across their length. The calculation is done by summing, sample by sample, the minimum distances between the signals by analysing the current and following sample between them and maintaining for the next comparison the index of sample with least distance. MFCC are obtained by taking the maximum amplitude of the cepstrum given by imposing in the spectrum triangular overlapping windows onto the Mel scale. MFCC provides a good method to extract features to classify phonemes of a speech. The Mel frequency cepstrum coefficients are calculated through the code provided at: `https://se.mathworks.com/matlabcentral/fileexchange/32849-htk-mfcc-matlab` over 70ms windows with 8ms of overlapping.

From the MFCC is excluded the first coefficient which represent the energy of the signal. For each audio it is calculated the DTW pairwise distances between its MFCC and the MFCC of the rest of the audio of the same digit and the MFCC of the audio generated from the middle point of the cluster of digits in the latent space:

Be x_{ij} the j-th instance in digit class $i \in \{0, 1, 2, \ldots, 9\}$ and X^i the set of all the instances of the class i. The vector of DTW pairwise distances pairwise_dtw$_{ij}$

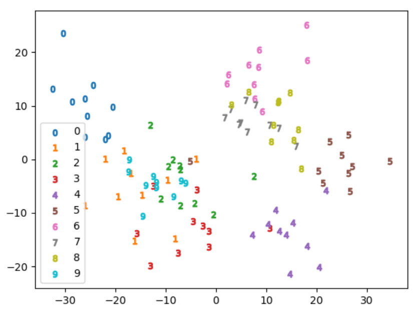
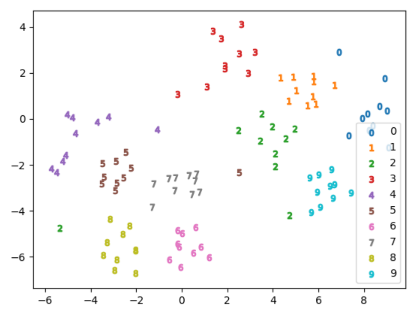
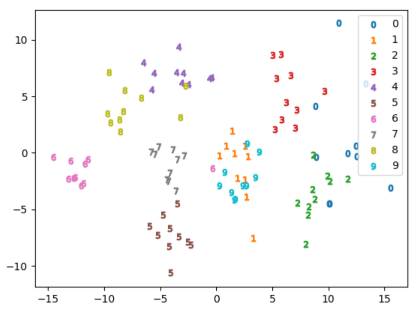
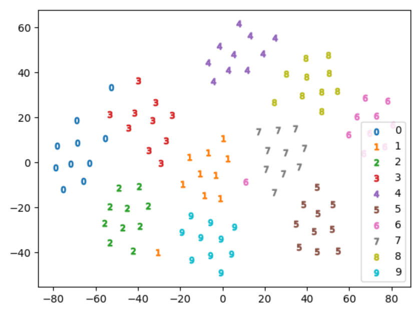
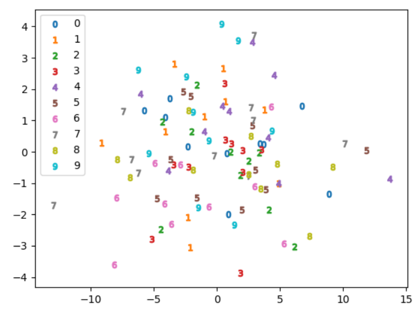
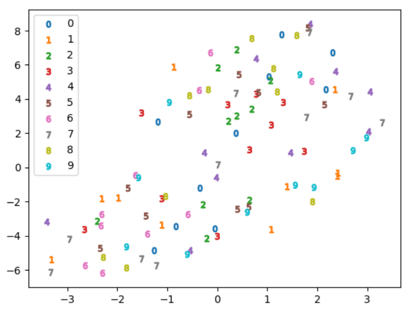
PCA projection:	t-SNE projection:
Convolution VAE 20 dim:	
	
Dilated VAE 20 dim:	
	
Hybrid VAE 20 dim:	
	

Table 5.3: Latent distribution projections for the spoken digits dataset over a 20 dimesnional latent space. Each row shows the results of the variational autoencoder based don the models introduced in sec. 4.1 implementing a classification loss at the bottleneck during the training, the left and the right columns show respectively the distribution accoridng to a PCA and t-SNE projections

is calculated between MFCC value mfcc(x_{ij}) and the vector of MFCCs mfcc($X^i \setminus \{x_{i,j}\}$) of the other samples of the same digit i excluding sample j:

$$\text{pairwise\_dtw}_{ij} = \text{dtw}(\text{mfcc}(x_{ij}), \text{mfcc}(X^i \setminus \{x_{ij}\})) \tag{5.1}$$

Be θ the encoder network and $z_{i,j} = \theta(x_{i,j})$ is its projection into the latent space, ψ the decoder mapping the latent sample $z_{i,j}$ back to the audio domain and Z^i the set of latent variables $Z^i = \theta(X^i)$. Then the DTW distance between the sample $x_{i,j}$ and the sound generated from the mean $\mu(Z^i)$ of the projections of digit i in the latent space

$$\text{center\_dtw}_{ij} = \text{dtw}(\text{mfcc}(x_{ij}), \text{mfcc}(\psi(\mu(Z^i)))) \tag{5.2}$$

The actual code used to calculate these is:

Matlab code: calculating DTW distances over MFCC:

```
for digit=0:9
    files = load_files_list_per_digit(digit);
    for k=1:length(files)
        [audio,fs] = audioread(files(k));
        audios(k,:) = audio';
        mfcc_cell{k} = mfcc(audio,fs,25,5,0.97,@hamming,[200 1200],20,13,22);
    end
    [middle,fs] = audioread(char(dir_mean+"mean"+digit+".wav"));
    mfcc_cell{11} = mfcc(middle,fs,25,5,0.97,@hamming,[200 1200],20,13,22);
    dtw_matrix = [];
    for k=1:10
        dtw_row = [];
        dtw_coeff = [];
        for j=k:10
            for i=1:13
                dtw_coeff(i) = dtw(mfcc_cell{k}(i,:),mfcc_cell{j}(i,:));
                end
            dtw_row(j) =  mean(dtw_coeff);
        end
        for i=1:13
            dtw_coeff(i) = dtw(mfcc_cell{k}(i,:),mfcc_cell{11}(i,:));
        end
        dtw_row(11) = mean(dtw_coeff);
        dtw_matrix = [dtw_matrix;dtw_row];
    end
    dtw_matrixes{digit+1} = dtw_matrix;
end
```

The actual dissimilarity distance between the generated audio from the cluster centre point and the distances distribution between one real audio and the rest of the element of the same digit is given by:

	x_0^1	x_1^1	x_2^1	x_3^1	x_4^1	x_5^1	x_6^1	x_7^1	x_8^1	x_9^1
x_0^1	0	384.67	106.06	106.74	114.45	112.78	143.42	117.92	70.66	119.71
x_1^1	384.67	0	341.84	366.413	346.69	376.20	433.46	381.67	396.40	377.98
x_2^1	106.06	341.84	0	87.24	53.98	98.62	143.23	105.83	100.13	96.27
x_3^1	106.74	366.41	87.24	0	93.42	65.35	121.32	87.59	107.85	85.10
x_4^1	114.45	346.69	53.98	93.42	0	108.88	134.89	114.18	105.49	106.21
x_5^1	112.78	376.20	98.62	65.35	108.88	0	116.41	61.66	110.24	64.15
x_6^1	143.42	433.42	143.23	121.32	134.89	116.41	0	139.95	123.05	125.13
x_7^1	117.92	381.66	105.83	87.59	114.18	61.66	139.96	0	109.19	85.19
x_8^1	70.66	396.40	100.13	107.85	105.49	110.24	123.05	109.20	0	125.78
x_9^1	119.71	377.98	96.27	85.10	106.21	64.15	125.13	85.19	125.78	0
a	141.82	378.36	125.91	124.56	130.91	123.81	164.53	133.69	138.75	131.73
b	83.51	355.91	91.17	86.96	101.85	85.81	126.97	88.14	87.37	101.42
c	-0.63	-0.83	-0.41	-0.41	-0.35	-0.39	-0.37	-0.47	-0.53	-0.32

Table 5.4: The table represent the results obtained from the cluster of the digit '1' for the dilation based variational autoencoder. The upper part shows the pairwise DTW distances of MFCC components for each audio in the testing set (see eq. 5.1). The a row is the mean of each column, the b row is the mean calculated in eq. 5.2, c is the represented dissimilarity between the audio generated form the centre of the cluster and the rest of the pairwise DTW distance among the audio in the testing set.

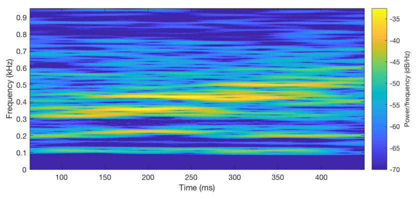
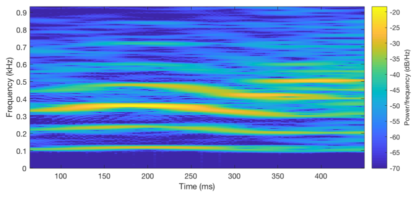
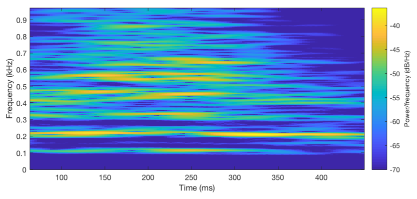
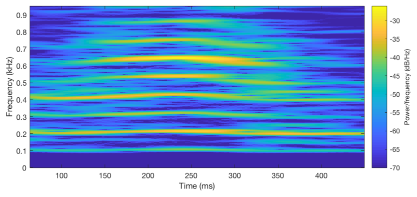
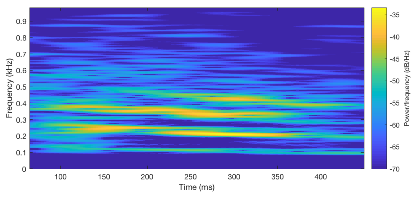
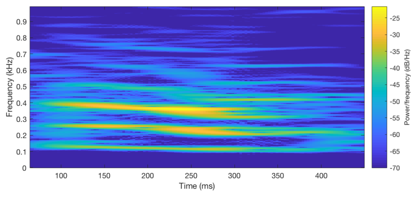
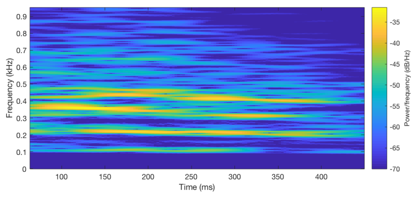
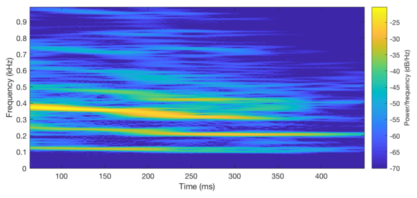
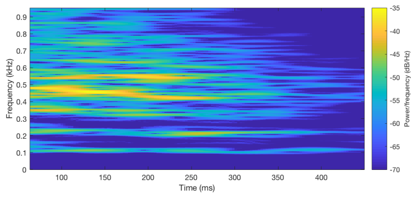
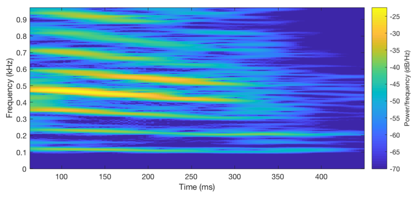
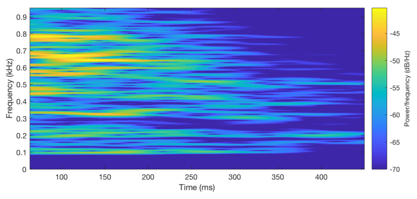
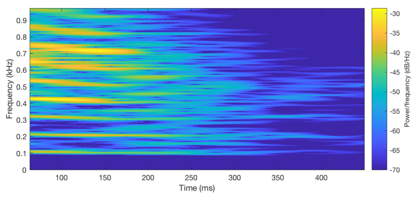
$$\text{dist}(x_{i,j}, \psi(\mu(Z^i))) = \frac{\text{center\_dtw}_{ij} - \mu(\text{pairwise\_dtw}_{ij})}{\sigma(\text{pairwise\_dtw}_{ij})} \tag{5.3}$$

For each digit class $i \in \{'0', '1', '2', ...'9'\}$, the mean and variance of $\text{dist}(x_{i,j} \in X^i, \psi(\mu(Z^i)))$, across all the audio instances in the i_{th} digit class is calculated. An example of complete distances table is shown in table 5.4 for the spoken digit '1'. The table 5.5 shows the results of the DTW similarity of the generated audio from the centre of each cluster compared to the DTW distribution of the real audio samples of the same cluster. This table shows the results of the mean and standard deviation from eq. 3.9 across all the dataset. We can observe that the mean is always non-positive for the dilation based model which means that the generated audio is closer in terms of MFCC to each of the original audio compared to the pairwise distances among them. The absolute value of the mean is always smaller than 1, which means that the audio generated by the geometrical centre of the region is actually also close to the centre in terms of MFCC similarities between all the audios. The results of the other two models follow the same trend from the 1-NN evaluation of the latent distribution. The convolution based model obtaines an average mean equal to 0.20, while the hybrid model gives largely bad results obtaining a drift equals to 5.47. It must be clarified that even if the generated audios refer to the centre point of the cluster, it is far from being just an average between all the audio, in which case it would be perceived as a chorus effect. The generated audio and the code are available at `https://github.com/mlionello/Audio-VAE`.

Table 5.6 shows, for completeness, the spectrograms of the generated audio from the middle point of each cluster besides the spectrograms of one sample audio used in the training set from the same class.

	DILA VAE:		CONV VAE:		HYBRID VAE:	
Digit cluster:	**Mean:**	**Std:**	**Mean:**	**Std:**	**Mean:**	**Std:**
zero	-0.29	0.96	0.79	1.65	6.89	2.47
one	-0.47	0.16	0.79	1,72	4.22	0.64
two	-0.38	0.51	0.03	1.13	4.27	0.86
three	0.00	1.19	0.69	1.71	5.36	1.67
four	-0.25	0.71	0.38	1.43	5.73	4.46
five	-0.46	0.14	0.02	1.14	7.58	1,63
six	-0.32	0.73	-0.24	0.93	5.71	1,38
seven	-0.37	0.27	-0.21	0.90	7.13	1,78
eight	-0.42	0.47	-0.05	1.19	4.96	1,81
nine	-0.32	0.68	-0.23	0.83	2.83	0.56
average:	-0.33	0.58	0.20	1.26	5.47	1.89

Table 5.5: The table shows the mean and standard deviation results from eq. 5.3 across all the dataset.

Digit:	**Generation:**	**Original:**
’0’		
’1’		
’2’		
’3’		
’4’		
’5’		

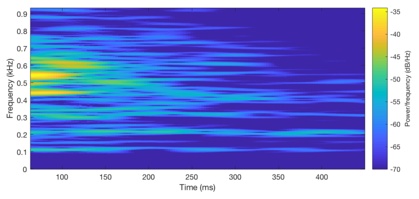
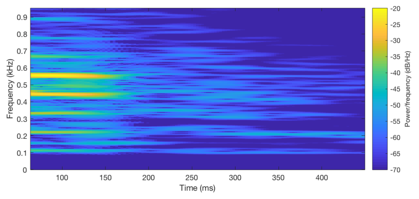
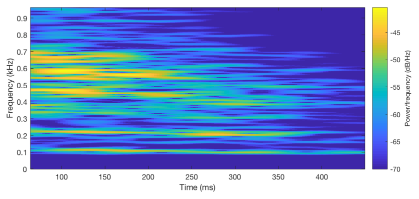
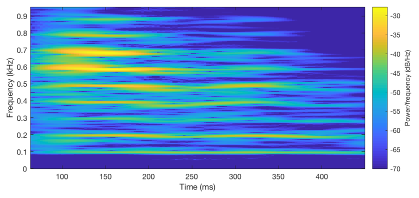
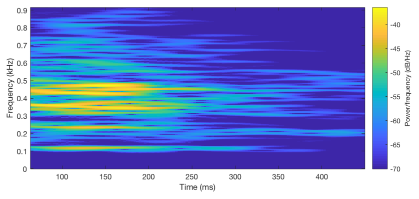
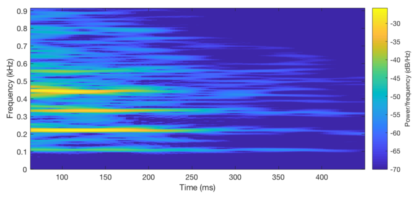
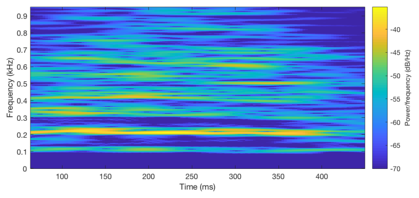
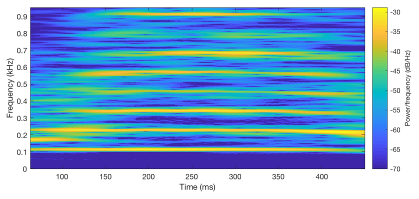
′6′		
′7′		
′8′		
′9′		

Table 5.6: Spectrogram of the generated audio from the middle point of each cluster of digits in the latent space and related original audio from one one of the sample used in the training set. The model used for these spectrograms is the dilation based variational autoencoder.

Chapter 6

Example of Application in Musical Context

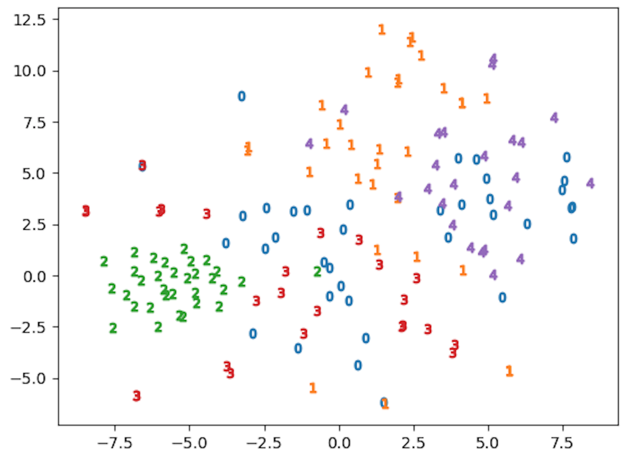


Figure 6.1: Scatter plot of the distribution of drums samples projected in the latent space. Bi-dimensional representation via t-SNE. Each number correspond to a different MFCC class.

The framework introduced in the previous chapter was tested in music domain, by feeding the variational autoencoder with a dateset of drums samples. The dataset comprehends 154 drum samples including kick, tom, hi-hat and snare sounds. Each audio is 16384 samples long, all with sample frequency of 22KHz. The samples have been labelled through a KMeans classifier applied to the average of MFCCs calculated on windows of 10ms samples covering the first 70ms of each of them. The MFCC returned 20 coefficients for each audio. A KMeans algorithm has been applied to the MFCCs in order to collect the audio in 5 different classes. The KMeans clustering methods was used to collect samples with similar acoustical features in the same region of the latent space. This procedure was motivated by the observation of the large variety of sounds obtained from the same drum instrument stored in the dataset.

The dataset presents multiple challenges. Each audio file is much longer than all the samples used in the previous chapters and the non-zero part for each sample varies significantly according to the drum sound. To avoid over fitting issues, the classification network at the bottleneck was reduced to 1 hidden layer of size equals to the number of clusters. The dimension of the hidden space is fixed to 10. The difference setting of parameters, with respect to what used for the digit spoken dataset, could be motivated because of the fact of the larger dissimilarities between the samples.

As seen in the t-SNE projection of the latent space in figure 6.1, the clustering does not reach good results except for one class of samples indicated with the symbol '2'. The symbols '1' occupy the top-right part of the projection, '3' are gathered in three main disconnected area and the '4' and the '0' overlap. Because of this the morphing between middle-point of the clusters has not been possible to be performed. Their audio generation and the audio of a morphing from 'hi-hat' to 'tom' are anyway still available at the github repository `https://github.com/mlionello/Audio-VAE`.

Figures 6.2, 6.3 and show the morphing in time and frequency domains between the sounds of a hi-hat and a tom. This is obtained by decoding the sampled path starting from the latent projection of one 'hi-hat' and ending at the coordinates of a 'tom'.

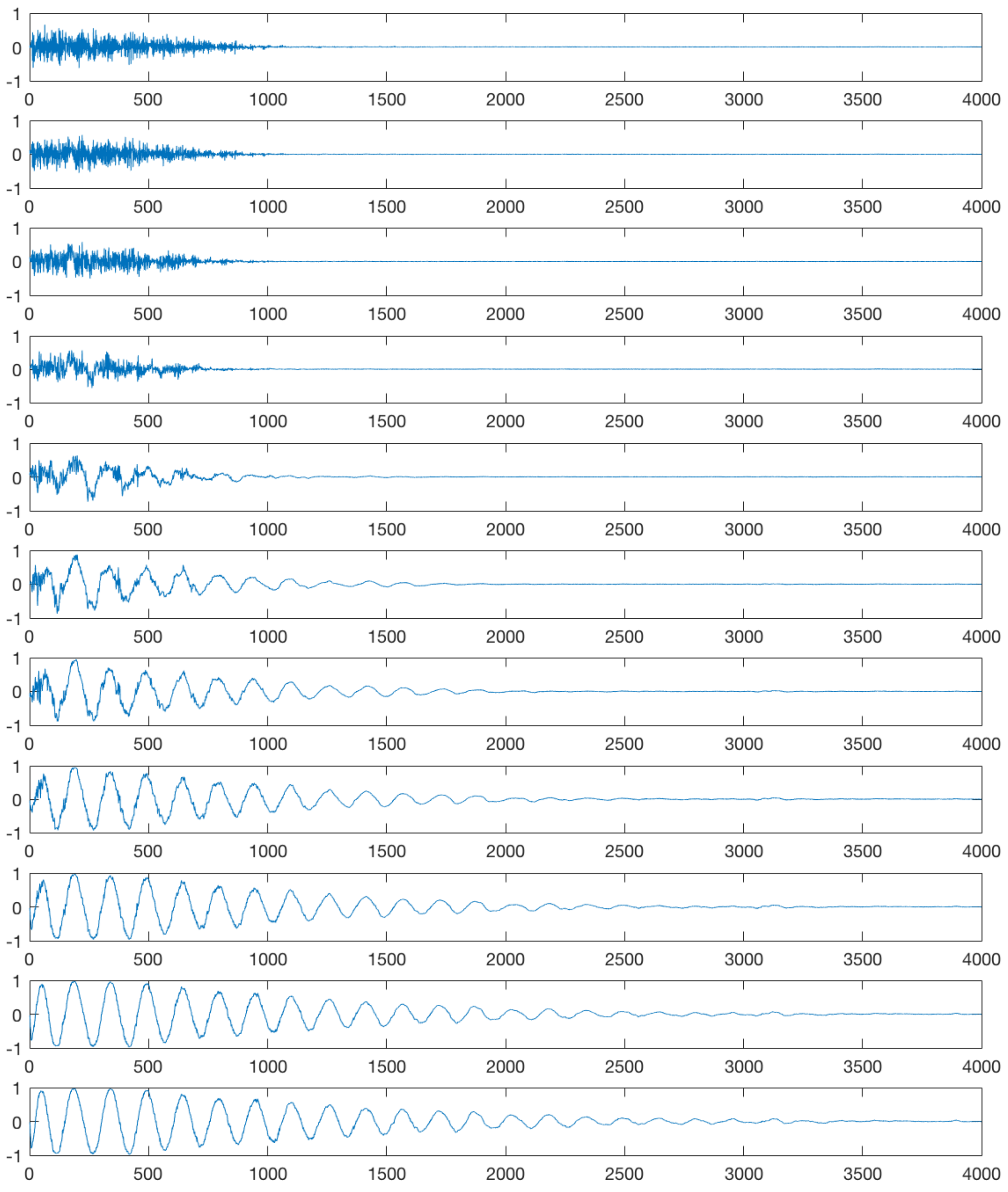


Figure 6.2: Morphing representation in time domain between an hi-hat and a tom sample

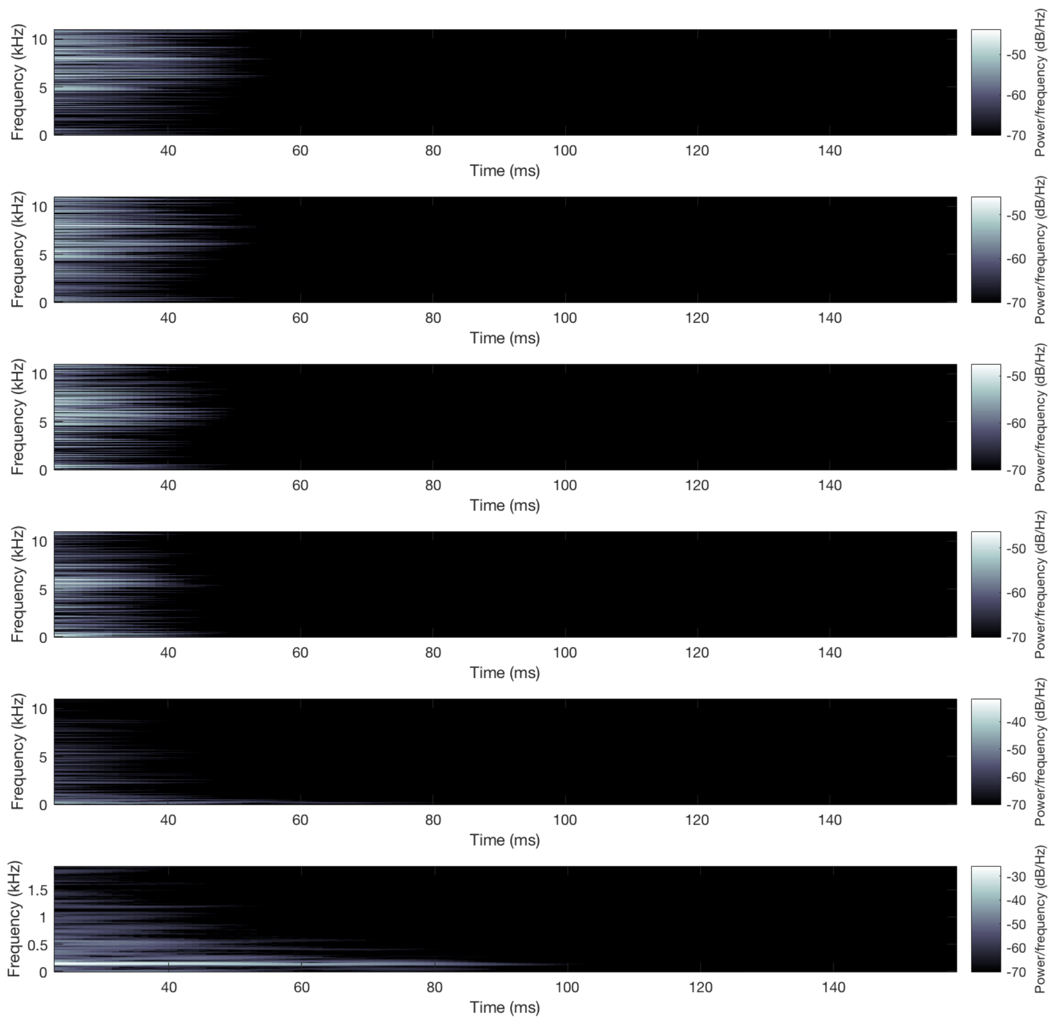


Figure 6.3: Morphing representation in frequency domain between an hi-hat and a tom sample, it continues in figure 6.4

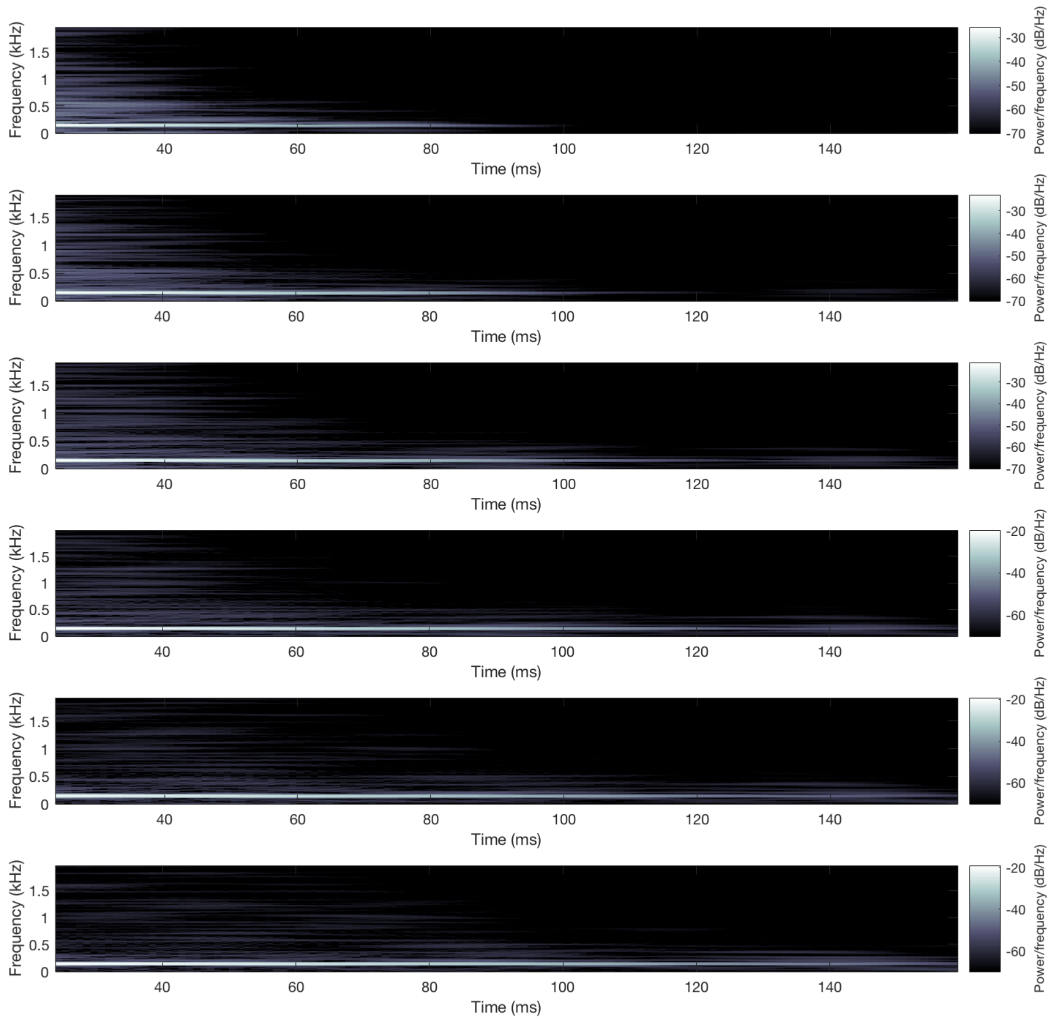


Figure 6.4: Second part of figure 6.3

Chapter 7

Conclusions and Future Work

This thesis presented a preliminary study on temporal-sequence representation of audio samples by means of autoencoder. Variational Autoencoders were then introduced to free this representation from temporal constrains. The obtained models were studied in order to distribute the extracted features according to the class of the object by introducing a second loss function to classify the latent representation in the bottleneck. The quality evaluation has been done by introducing a Dynamic Time Wrapping algorithm comparing the Mel Frequency Cepstrum Coefficients distances among the training data and its distribution over one generated sample and the original ones. The experiments were conducted on a dataset of spoken digits. They show positive results for the latent distribution and for the generation task. It has been seen that Variational Autoencoders based on dilation layers, performed better results with respect to convolution layers. A combination of these two approaches appears to be more complicate to be trained because of the large amount of parameters introduced by their concatenation.

However, some topics were left to be further investigated in the context of this thesis. The first point to be further investigated is to find a good set of parameters for the drum dataset to perform a better clustering over the latent space. The second topic is to understand local distributions in the latent space by investigating according to what criteria audio samples are organised in the same sub-region. A third topic concerns to extend the latent space to one more dimension, reintroducing the temporal information for generate samples by flowing their representation across the extra dimension and jumping among their neighbours at the same time. A last point to be implemented is a fine-tuning for the model by means of Generative Adversarial Network. This implementation, trained over random generated point from the probability space with respect to unpaired original audio from the same sub-region of the latent space, will allow to reach a better output quality of generated audio across the whole latent space, especially for morphing tasks.

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Uwe Andresen. "A New Way in Sound Synthesis". In: *Audio Engineering Society Convention 62*. 1979. URL: http://www.aes.org/e-lib/browse.cfm?elib=2920.

[3] Jose Luis Diez Antich. "Audio Event Classification using Deep Learning in an End-to-End Approach". MA thesis. Copenhagen, Denrmak: Department of Architecture, Design, and Media Technology, Aalborg University Copenhagen, 2017.

[4] Jose Bernardo et al. "Generative or Discriminative? Getting the Best of Both Worlds". In: 8 (Jan. 2007), pp. 3–24.

[5] Charles Brazier. *Machine Learning Methods for Music Transformation*. Aalborg University Copenhagen, 2017. URL: goo.gl/d1PDLz.

[6] John M. Chowning. "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation". In: *J. Audio Eng. Soc* 21.7 (1973), pp. 526–534. URL: http://www.aes.org/e-lib/browse.cfm?elib=1954.

[7] C. Doersch. "Tutorial on Variational Autoencoders". In: *ArXiv e-prints* (June 2016). arXiv: 1606.05908 [stat.ML].

[8] Chris Donahue, Julian McAuley, and Miller Puckette. "Synthesizing Audio with Generative Adversarial Networks". In: *CoRR* abs/1802.04208 (2018). arXiv: 1802.04208. URL: http://arxiv.org/abs/1802.04208.

[9] Jesse Engel et al. "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders". In: *CoRR* abs/1704.01279 (2017). arXiv: 1704.01279. URL: http://arxiv.org/abs/1704.01279.

[10] Gajhede, Nicolai, Beck, Oliver, and Purwins, Hendrik. "Convolutional Neural Networks with Batch Normalization for Classifying Hi-hat, Snare, and Bass Percussion Sound Samples". In: (2016).

[11] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

[12] Nitin Goyal and R.K. Purwar. “Analyzing Mel Frequency Cepstral Coefficient for Recognition of Isolated English Word using DTW Matching”. In: *IJRCCT* 3.4 (2014).

[13] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[14] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory”. In: 9 (Dec. 1997), pp. 1735–80.

[15] Daniel Jiwoong Im et al. “Denoising Criterion for Variational Auto-Encoding Framework.” In: *AAAI*. 2017, pp. 2059–2065.

[16] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *CoRR* abs/1611.07004 (2016). arXiv: `1611.07004`. URL: `http://arxiv.org/abs/1611.07004`.

[17] William James. *The Principles of Psychology*. Dover Publications, 1890.

[18] D. Jimenez Rezende, S. Mohamed, and D. Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: *ArXiv e-prints* (Jan. 2014). arXiv: `1401.4082 [stat.ML]`.

[19] D. P Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *ArXiv e-prints* (Dec. 2013). arXiv: `1312.6114 [stat.ML]`.

[20] Volodymyr Kuleshov, S Zayd Enam, and Stefano Ermon. “Audio Super Resolution using Neural Networks”. In: *arXiv preprint arXiv:1708.00853* (2017).

[21] Anders Boesen Lindbo Larsen et al. “Autoencoding beyond pixels using a learned similarity metric”. In: *arXiv preprint arXiv:1512.09300* (2015).

[22] Xiang Li et al. “Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift”. In: *CoRR* abs/1801.05134 (2018). arXiv: `1801.05134`. URL: `http://arxiv.org/abs/1801.05134`.

[23] H. Purwins M. Abou-Zleikha M. Lionello L. Pietrogrande. “Accepted for publication - Exploration of Musical Space with Parametric t-SNE in a Browser Interface”. In: *Sound and Music Computing Conference*. 2018.

[24] R. Ramirez M. Lionello. “Accepted for publication - A Machine Learning Approach to Violin Vibrato Modelling in Audio Performances and a Didactic Application for Mobile Devices”. In: *Sound and Music Computing Conference*. 2018.

[25] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[26] Andrea Corcuera Marruffo. “Automatic sonification of video sequences through object detection and physical modelling”. MA thesis. Copenhagen, Denrmak: Department of Architecture, Design, and Media Technology, Aalborg University Copenhagen, 2017.

[27] Minsky M. Rochester N. Shannon C.E. McCarthy J. *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. 1955.

[28] W. Pitts McCulloch W. S. “A Logical Calculus of Ideas Immanent in Nervous Activity”. In: *Bulletin of Mathematical Biophysics*. Vol. 5. 1943, pp. 115–133.

[29] P. Mermelstein. “Distance measures for speech recognition, psychological and instrumental”. In: *Pattern Recognition and Artificial Intelligence*. Ed. by C. H. Chen. New York: Academic Press, 1976, 374–388.

[30] Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. “Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks”. In: *CoRR* abs/1701.04722 (2017). arXiv: 1701.04722. URL: http://arxiv.org/abs/1701.04722.

[31] Bhadragiri Jagan Mohan and Ramesh Babu N. “Speech recognition using MFCC and DTW”. In: *2014 International Conference on Advances in Electrical Engineering (ICAEE)*. 2014, pp. 1–4. DOI: 10.1109/ICAEE.2014.6838564.

[32] Jagan Mohan. B and Ramesh Babu.N. “Speech recognition using MFCC and DTW”. In: (Jan. 2014).

[33] N. Mor et al. “A Universal Music Translation Network”. In: *ArXiv e-prints* (May 2018). arXiv: 1805.07848 [cs.SD].

[34] Lindasalwa Muda, Mumtaj Begam, and I. Elamvazuthi. “Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques”. In: *CoRR* abs/1003.4083 (2010). arXiv: 1003.4083. URL: http://arxiv.org/abs/1003.4083.

[35] Andrew Y. Ng and Michael I. Jordan. “On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes”. In: *Advances in Neural Information Processing Systems 14*. Ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, 2002, pp. 841–848. URL: http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes.pdf.

[36] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel Recurrent Neural Networks”. In: *CoRR* abs/1601.06759 (2016). arXiv: 1601.06759. URL: http://arxiv.org/abs/1601.06759.

[37] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. "Neural Discrete Representation Learning". In: *CoRR* abs/1711.00937 (2017). arXiv: `1711.00937`. URL: `http://arxiv.org/abs/1711.00937`.

[38] Aäron van den Oord et al. "Conditional Image Generation with PixelCNN Decoders". In: *CoRR* abs/1606.05328 (2016). arXiv: `1606.05328`. URL: `http://arxiv.org/abs/1606.05328`.

[39] Aäron van den Oord et al. "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *CoRR* abs/1711.10433 (2017). arXiv: `1711.10433`. URL: `http://arxiv.org/abs/1711.10433`.

[40] Mattia Paterna. "Timbre modification using deep learning". MA thesis. Copenhagen, Denrmak: Department of Architecture, Design, and Media Technology, Aalborg University Copenhagen, 2017.

[41] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. "Large-scale Deep Unsupervised Learning Using Graphics Processors". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: ACM, 2009, pp. 873–880. ISBN: 978-1-60558-516-1. DOI: `10.1145/1553374.1553486`. URL: `http://doi.acm.org/10.1145/1553374.1553486`.

[42] R. Ranganath et al. "Operator Variational Inference". In: *ArXiv e-prints* (Oct. 2016). arXiv: `1610.09033 [stat.ML]`.

[43] F. Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review* (1958), pp. 65–386.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1". In: ed. by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning Internal Representations by Error Propagation, pp. 318–362. ISBN: 0-262-68053-X. URL: `http://dl.acm.org/citation.cfm?id=104279.104293`.

[45] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49. ISSN: 0096-3518. DOI: `10.1109/TASSP.1978.1163055`.

[46] Jan Schluter and Sebastian Bock. "Improved musical onset detection with convolutional neural networks". In: *Acoustics, speech and signal processing (icassp), 2014 ieee international conference on*. IEEE, 2014, pp. 6979–6983.

[47] Jan Schluter, Karen Ullrich, and Thomas Grill. "Structural segmentation with convolutional neural networks mirex submission". In: *Tenth running of the Music Information Retrieval Evaluation eXchange (MIREX)* (2014).

[48] Franciska de Jong Thijs Westerveld Arjen de Vries. “Generative Probabilistic Models”. In: (2007), pp. 177–198.

[49] Tomas Gajarsky, Hendrik Purwins. “An Xception Residual Recurrent Neural Network forAudio Event Detection and Tagging”. In: *Sound and Music Computing Conference*. 2018.

[50] A. van den Oord et al. “WaveNet: A Generative Model for Raw Audio”. In: *ArXiv e-prints* (Sept. 2016). arXiv: `1609.03499 [cs.SD]`.

[51] REN Li-mian WEI Ming-zhe LI Xi. “Improved DTW Speech Recognition Algorithm Based on the MEL Frequency Cepstral Coefficients”. In: *Information and Communication Technology and Smart Grid*. 2010.

[52] B. Widrow. “An adaptive ’Adaline’ neuron using chemical ’memistors’”. In: (1960).

[53] Yonghui Wu et al. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: *CoRR* abs/1609.08144 (2016). arXiv: `1609.08144`. URL: `http://arxiv.org/abs/1609.08144`.

[54] Iannis Xenakis. *Formalized Music Thought and Mathematics in Composition*. Indiana University Press, 1971.

[55] Kelvin Xu et al. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International Conference on Machine Learning*. 2015, pp. 2048–2057.

[56] Yong Xu et al. “Fully Deep Neural Networks Incorporating Unsupervised Feature Learning for Audio Tagging”. In: *CoRR* abs/1607.03681 (2016). arXiv: `1607.03681`. URL: `http://arxiv.org/abs/1607.03681`.

[57] Yassine Mhiri. *An Overview of Sound Generation Using Generative Adversarial Neural Networks*. Aalborg University Copenhagen, 2017. URL: `goo.gl/ga7UhF`.

[58] Shuchang Zhou et al. “GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data”. In: *CoRR* abs/1705.04932 (2017). arXiv: `1705.04932`. URL: `http://arxiv.org/abs/1705.04932`.

[59] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *CoRR* abs/1703.10593 (2017). arXiv: `1703.10593`. URL: `http://arxiv.org/abs/1703.10593`.