MASTER’S THESIS IN COMPLEX ADAPTIVE SYSTEMS,
DEPARTMENT OF PHYSICS
GOTHENBURG , SWEDEN 2022

Music Audio Signal Prediction using Machine Learning

Ivan Gentile

CHALMERS
UNIVERSITY OF TECHNOLOGY

musictribe

Abstract

Even though considerable advancements have been made in time series forecasting for audio, there are still many unexplored aspects. An objective of the analysis is to develop a viable product to replace the look-ahead functions of audio dynamic range compressors. Towards this end, and given the suitability of neural networks for predictive purposes, this project discusses the application of MultyLayer Perceptrons (MLPs) and Long-Short Term Memory (LSTMs) for addressing this research question. The numerical experiments focuses on the predictions of this systems. It is analyzed how changing window length (number of inputs), prediction steps (number of outputs), and sampling frequency (dataset resolution) affects prediction quality. The findings indicate that, after a threshold, increasing number of inputs yields diminishing rewards.

Keywords

Audio Digital Signal Processing, Machine Learning, Artificial Intelligence, Time Series Forecasting, Audio Machine Learning Application, Signal Prediction, Time Samples Prediction

Acknowledgements

Time appears to flow only forward. Life as an individual would be subject to this constraint, if it were not for human relationships. I see connections between people as being able to go beyond our intuition about time and entangle past, present and future in a really intricate and marvellous web.

This work marks a critical point in my life. It is with immense gratitude that I would like to dedicate it to all my friends and families who accompanied me to it. I name just a few special ones: Agnorelli, Angelov, Appert, Barry, Bucchieri, Conti, Crivaro, Douros, Eberle, Esposito, Ferlito, Garcia, Ghidotti, Gruszczyńska, Leo, Lisbon, Marinov, Marsal Ortin, Mauri, Müller, Molesini, Mornati, Palmisano (Rock&Roll) Perolari, Rizzo, Roselli, Ruggi, Sala Peup, Simeonov, Tan, Vasiluta, Vergalli.

An extraordinary thanks go to the Macaluso Green family who I have joined for one year and I have felt like an honorary member every since. Without Jeff and Jenny's help, I would not be writing these words at this moment. GRAZIE MILLE!

Thank you to my own family: my mum, my brother, my sister, my aunt and my two cousins. I can only tell you that the more I live, the more I am proud of being Gentile. Vi voglio bene, e da buoni discepoli di nonna sapete cosa significhi.

Thanks to my sweet half, Aurora, for giving flavour to my days. Non posso immaginare niente di meglio di averti accanto.

I would like to thank Jesper Pedersen and Gustav Santesson for envisioning this project, allowing me to partake in it and providing me with extensive support for the whole project duration. Thanks also to Sacha Krstulovic and Spyros Stasis who joined us during the path. I appreciate a lot the climate of our meetings and all the lessons you gave me.
Finally, thanks to Mats Granath for his encouraging supervision.

Author

Ivan Gentile <ivogentile@hotmail.it>
Physics Department
Chalmers University of Technology

Work performed in:

Kungsbacka, Sweden
Milan, Italy

Examiner

Mats Granath
Gothenburg, Sweden
Chalmers University of Technology

Supervisors

Jesper Pedersen
Kungsbacka, Sweden
MusicTribe Sweden

Gustav Santesson
Kungsbacka, Sweden
MusicTribe Sweden

Sacha Krstulovic
Manchester, UK
MusicTribe

Spyros Stasis
Manchester, UK
MusicTribe

Carlo Bolla

Manchester, UK

MusicTribe

Foreword

It was 1885 when, thanks to the initiative of Gösta Mittag-Leffler, a mathematics competition in the name of Oscar II, king of Sweden and Norway, was published in the journal ”Acta Mathematica” [1]. As part of the competition, contestants were asked to find a general solution to the classical mechanics n-body problem, something that mathematicians have been hunting for since Newton’s time. This competition turned into history thanks to the solution provided by Henri Poincaré [2], which not only granted him the 2,500 Swedish kronor and gold medal prize but contained concepts that laid the foundations for the new field of Chaos Theory.

More than one hundred and thirty years later, I have started grappling with the bizarre and beautiful ideas that such theory exposes through my studies on Complex Adaptive Systems[1] and I am in this work exposing the potentiality of employing such ideas through a project that MusicTribe proposed. The discussion that follows these lines will most often be technical and pragmatic and can lead one to quickly forget about the importance of the theoretical work done in the past by great intellects, through stories such as the one mentioned above on Poincaré. Nevertheless, I encourage you to recognize that we will be all along resting on the shoulder of these giants, all along discussing the power of analytical ideas. Only we will see these abstract general ideas manifest in the spirit of today, incarnated in the incredible tools that our information era provides us.

A perspective on Machine Learning

It is of my interest to also address a general concern related to the Artificial Intelligence (AI) field. To do so let us start with a quote from Pedro Domingos, professor emeritus of computer science and engineering at the University of Washington

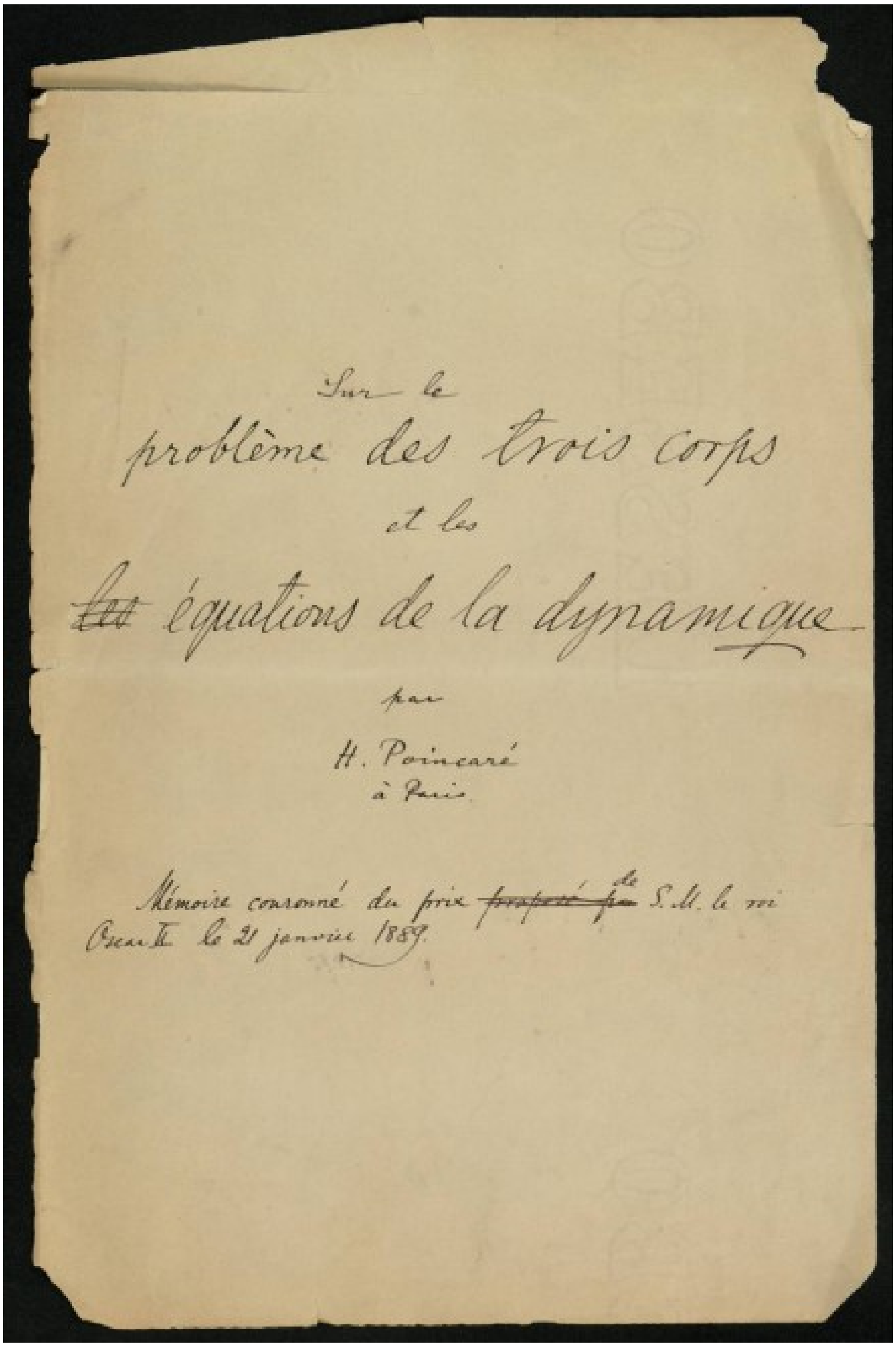
People worry that computers will get too smart and take over the

[1]The knowledge of emergence and collective behaviour is the linchpin of Complex Systems and very much rest upon insights gathered through the observance of Chaos

> world, but the real problem is that they're too stupid and they've already taken over the world.

From this we pick up not only a characterization of AI and its dangers which also readers foreign to the field have been exposed to[2], but also a clear statement regarding the functioning of these algorithms: Machines learn stupidly, not smartly. Attempting to reconcile this fact in order to make an intelligent product has been my occupation for the duration of the internship. Even if this target has not been reached, I hope that what follows will provide you useful insights, back-propagating my efforts to an improvement of your work.

[2]Indeed many popular figures, including Sam Harris (URL) and Elon Musk (URL), are warning the general public about the existential danger of AI, while a more appropriate view would be the one exposed by AI experts such as Janelle Shane (URL)



Sur le

problème des trois corps

et les

~~les~~ équations de la dynamique

par

H. Poincaré

à Paris

Mémoire couronné du prix ~~proposé par~~ de S. M. le roi Oscar II le 21 janvier 1889.

Figure 0.1: Front cover of the original manuscript forwarded by Poincaré to Acta Mathematica containing a dynamical solution to the three body problem. Copyright to Institut Mittag-Leffler

Contents

1 Introduction

After listening to a song for a few times, many of us find it easy to replay the song in our head, singing along the lyrics and humming its rhythm. Sometimes it is even sufficient to listen to only parts of the song, in order to be able to anticipate what the next chorus will sound like. Given enough training, our brain seems then capable of abstracting the patterns present in music in order to predict what the next part of the track will sound like. Can an information computing system residing on semi-conductor materials perform a similar task?

This thesis deals with music audio signal prediction. Machine Learning (ML) techniques, applied mainly within a Time Series Forecasting (TSF) framework, are used to obtain such predictions. The primary objective of this research is to investigate the performance quality of supervised learning algorithms applied to raw audio music data. The investigation aims to find an algorithm that can process a given sequence of audio track time samples and output a sequence that matches the original sequence. Here a time sample is a real value that corresponds to the amplitude of the waveform of the audio signals.

The questions at the core of the research are:

- Is it possible to feed a Neural Network (NN) raw musical audio data and obtain a valid forecast of the evolution of such data for a short time window?
- Which supervised learning method is best at performing such a task?
- Is it possible to obtain such NN for a use scenario that ranges over on different kinds of musical genres and instruments or is it necessary to restrict its application to a particular kind of genre or musical instruments?

Answering these questions is crucial for the application of these NNs in the audio music industry. Indeed, a use case for a well-performing neural network is already envisioned; Applying a forecasting algorithm to music audio tracks would be valuable as a pre-processing tool for audio equipment. In particular dynamic range compressors could benefit from its use by eliminating the need of a look-ahead function, hence decreasing the latency of the audio system.

1.1 Background - Machine Learning Applied to Audio

Living in 2022, it is conceivable to recognize that a ”AI spring” is in blossom [3]. The current success of Artificial Intelligence and Machine Learning (ML), ranging from Language Translation (Google Translate) to Natural Language Generation (GPT-3), from Image Recognition (ImageNet) to game playing systems (Alpha GO) ferments a wave of optimism on the possibility of ML succeeding in a wide range of applications.[3]

It comes as no surprise that ML approaches are also blooming in the Audio Domain. To list a few of the applications of ML in this sector we can recall: Audio Classification [4], Speech Synthesis [5], Sound Event Detection [6], Audio Source Separation [7], Audio Signal Processing [8] and others. It appears, when examining the audio domain and other fields, that when applied to tasks for which adequate data is available, machine learning algorithms tend to outperform traditional methods previously employed. For example, ML algorithms have proven to be better performers for some use cases in Audio Signal Processing compared to classic methods such as Gaussian mixture models, hidden Markov models and non-negative matrix factorization [9]. This view justifies the utility in testing whether a novel application of ML can replace the traditional use of a look-ahead function within dynamic range compressors. Given such hypothesis, a successful replacement would be indicated by an overall lower audio latency of the audio system caused by the instrument. This also puts constraints on the complexity of the ML Technique deployed.

Here it is not pertinent to focus on the technical details behind the effectiveness of the ML procedures in the aforementioned cases. It will suffice to be aware of the large extent to which AI is present within the Audio and Sound Engineering fields. Hence it is not possible to provide a comprehensive description of AI's applications in the audio domain. Yet, this works provide a look on a subset of supervised learning techniques, namely MultiLayer Perceptrons (MLPs) and Recurrent Neural Networks (RNNs). Integrating these ML architectures, the raw

[3]One example of possible counterargument regarding the overarching optimism might be provided by the delay in circulation of Self Driving Cars.

audio music data, and the analysis under a TSF framework is the innovation and primary focus of this thesis. This is the key addition of this study, as there is comparatively much less research on TSF applied to audio data compared to the amount of literature that employs ML for speech synthesis, voice-to-text conversion, and genre categorization.

To get insight from this project, just minimal prior knowledge is necessary. The part on theoretical background that follows should offer sufficient information on ML approaches, Time Series Forecasting and audio data properties for an engineering-savvy reader to follow the rest of the project.

1.2 Project Description

This study aims to investigate how ML may be used to enhance the prediction of audio samples. The work takes place in the context of an internship at MusicTribe in Kungsbacka, Sweden. Therefore, an effort is made to conceive of a machine learning tool that may be directly applied to company products. As mentioned above, the main use considered for such ML tool is replacing the compressor look-ahead control, enabling an overall lower latency caused by the compressor instrument. In spite of this, the applicability of an effective audio prediction ML method extends beyond look-ahead replacement. In fact, given the plasticity of NNs, it is conceivable to envision future uses of such algorithms in a variety of audio equipment.

For the whole duration of the project, numerical experiments have been performed through the use of the Python programming language and the TensorFlow ML library. The training and testing of the algorithms were carried out using the personal laptop of the author, operating with a AMD Ryzen 7 5800H CPU and NVDIA GeForce RTX 3070 Laptop GPU. Most of the algorithms were optimized for GPU performance using CUDA toolkit and CUDNN library, allowing parallel computing.

1.3 Thesis Outline

The structure of this thesis is as follow:
In Chapter 2 the theoretical background comprising an overview of the machine learning techniques employed, necessary knowledge pertaining to the analysis of audio data and basic notions of time series forecasting is provided.
In Chapter 3 the methodology behind the research is illustrated following a practical point of view.
In Chapter 4 a description of the different stages of research and experience within the company is provided.
In Chapter 5 the results of the analysis are presented.
In chapter 6 the major contributions stemming from such results are discussed.

2 Theoretical Background

The ultimate goal of the research is to provide an algorithm which is able to process a given sequence of n_t time samples of an audio track and output a generated sequence p_{t+} of subsequent time samples which closely matches the real sequence of the audio signal n_{t+}. Here a time sample, n_0, is a real value that corresponds to the amplitude of the waveform of the audio signals at time $t = 0$. Given this explanation, it should be apparent why the majority of the analysis given will be framed as a time series forecasting exercise. The goal of this section is three-fold:

- Introduce the reader to ML concepts
- Introduce the reader to basic properties of audio data.
- Provide an overview on the field of Time Series Forecasting.

2.1 An introduction to Machine Learning

To start, let us examine a clear definition of Machine Learning[10]:

> ML is an algorithmic area that combines concepts from statistics, computer science, and several other fields to build algorithms that analyze data, generate predictions, and aid in decision-making.

Given that ML is regarded as an applied science, it seems reasonable to start the topic by introducing the reader to fundamental statistical concepts using linear algebra. Introducing these topics is beyond the scope of this project, for further information please refer to [11]. This research focuses instead on the subfield of machine learning known as supervised learning. This implies assuming the reader is familiar with the fact that a ML model is comprised by weights and connections between this weights which form a causal chain from input to output-prediction.

ML methods are characterized by two factors: the data used for learning and the optimization method which enables the machine to learn. Let us start by describing the process of optimization. First of all, it is important to understand why is it called optimization. From the standpoint of mathematics and computer

science, optimization is the selection of the optimal element from a collection of possibilities, based on some criterion[12]. Usually this is the search for minimum or maximum of a real function. Indeed, training refers the process of iterating a set of computations in order to reach the minimum of a loss function.
Using an analogy with the real world, whereby the aforementioned real functions refers to the steepness of a hill, and the optimization process refers to reach the bottom of such hill, one should understand that in order to find the minimum the requirements are first to understand in which directions one should walk, and second to actually take steps in that direction. Continuing the analogy it is possible to imagine that in the ML model the entity walking down (stochastically, hence blind-folded) a high-dimensional hill[4]. This analogy provides an intuition for the algorithmic procedure which leads a model to improve predictions after training, by finding the gradient of the loss function (the direction towards the minimum) and by updating its weight accordingly (walking in such directions).
The machinery to calculate the gradients of a network is called Back-propagation[5] [13]. Some of the most popular optimizer options include Stochastic Gradient Descent [14], RMSProp [15],ADAM[16]

For completeness notice that unsupervised learning, in which transformations of input data are discovered without the need for target labels. The case of weakly supervised learning [17] is likewise intriguing for the task at hand, but it is beyond the scope of this work.

2.1.1 Supervised Learning

Essentially, supervised learning forces neural networks to learn by example, by finding a mapping between the given input data and target labels which are provided. Labeled data is the defining characteristic of Supervised Learning. This scenario is an ideal fit for the work at hand, as the audio track naturally offers the required labels, namely the signal waveform amplitude (see Figure 2.1).

[4]The dimension space of the search is dependent upon the number of inputs provided to the net.

[5]Technically this term is appropriate only for feed-forward networks (such as MLPs) and not for other models (such LSTMs), yet it is common to refer to different methods for updating model weights as such.

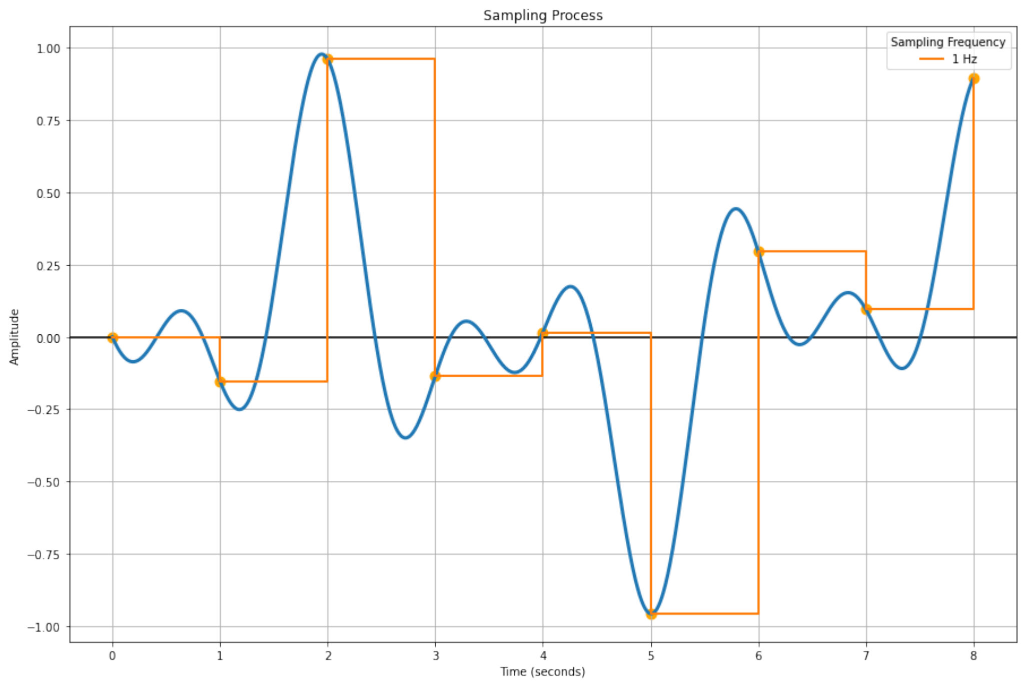


Figure 2.1: A visualization of the sampling process. In blue it is pictured a simple waveform. In orange it is showed the sampled signal with sampling frequency of 1 Hz. The sample points (orange dots) will work both as input and as target labels for the net in question. An example of a regression task for the network will then be, given the first four sample points predict the fifth sample point. A success for the net would be a prediction close to the target value, i.e. having a prediction of 0 for the amplitude of the fifth time sample would be considered satisfactory.

Supervised learning is by far the most common among different branches of ML [13]. The nets find correlations between the given parameters, so establishing a cause-and-effect link between the variables and the targets in the dataset. Here it is important to notice that a partition of the dataset is made in order to efficiently carry on a ML study.
First a training dataset is necessary. This is in essence the learning material for the ML method. Once the training set has been given to the model, it is able to undertake optimization and back-propagation, which, based on the characteristics of the training dataset and the selected loss function, will result in improved predictions.
The fitted model is then used to predict the answers for the observations in a second dataset known as the validation data set. The validation data set gives an objective assessment of a model's fit to the training data set while hyperparameters are being tuned [18].

Lastly, the test dataset is a dataset used to offer an impartial assessment of the model's final fit to the training data set.

Typically, supervised learning is used in applications where previous data is used to anticipate probable future occurrences. Supervised learning is also commonly used for regression task and for these reasons it was considered the optimal choice for the project.

2.2 Machine Learning Architectures

The three major types of network architectures are: dense networks, convolutional networks, and recurrent networks. Regardless of the branch of ML used for training and testing, each type of network is designed for a particular input mode: a network's architecture (dense, convolutional, recurrent) encodes assumptions about the structure of the data, to learn the structure space and correctly predict new inputs. Compatibility between the dataset and the network architecture's fundamental principles determines whether a specific design will function in a certain context[13].
An important concept which is necessary to discuss is that of the activation function. Essentially, an activation function is an essential infredient to be able to represent non-linear functions. Some of the most popular activation function include ReLU [6], sigmoid, tanh, Softmax [19].

In this study, music audio data cast as time series data were tested on both densely connected and recurrent networks. Below is a full discussion of the types of networks utilized.

2.2.1 Densely connected networks - MultiLayer Perceptron

A densely connected network consists of a series of Dense layers that are mostly utilized for vector data processing (batches of vectors). Such networks assume no particular input pattern; they are called densely connected because each neuron in a Dense layer is connected to every other neuron in the next layer. This form of network is the direct successor of Perceptrons, the first type of artificial

[6]see this article

intelligence that was investigated extensively by Rosenblatt in 1957 [20]. For this reason, this study refers to a stack of dense layers as MultiLayer Perceptrons.

2.2.2 RNNs - Long Short Term Memory

RNNs function by processing input sequences one time step at a time and maintaining a state throughout[7]. In the case of time series data, when the sequences of patterns of interest are not invariant over temporal translation, their application is widespread. RNNs may discriminate between various temporal sequences by attributing a high predictive power to values near to the required prediction while still considering the effect of values from the distant past..
The Long Short Term Memory (LSTM) algorithm was developed in 1997 in order to avoid a problem with standard RNNs topologies, which tend to suffer of vanishing gradient problem[21]

2.2.3 Comparison LSTM vs MLP

While feedforward networks, such as MLPs, have the capabilities to learn any nonlinear function, and hence are suitable for the task at hand, recurrent neural networks are usually preferred for uses cases that involve TSF.
The two main advantages of recurrent networks are:

- RNN captures the sequential information present in the input data
- RNNs share parameters over multiple time steps. This is commonly referred to as Parameter Sharing. This reduces the number of parameters to train and the computational cost.

2.3 Basic properties of Audio Data

Despite the kind of audio used in order to conduct the analysis (studio audio track or live recordings, of a specific musical genre or of mixed genres, single instrument outputs or even pure tones such as square or sine waves) this data consists of Continuous Time (CT) signals. Yet the input for the NNs needs to be Discrete. This poses the problem of transferring a CT to a DT signal, and this common reduction

[7]a state is typically a vector or set of vectors: a point in a geometric space of states

problem takes the name of sampling.

2.3.1 Audio Sampling

The transformation from Continuous to Discrete time is usually performed by an encoding of a subset of integers $I \subset Z$ over the continuous interval. Hence the Discrete Time signal gets defined as a function $x : I \to R$, where the domain I corresponds to points in time.The most used sampling method to translate a CT signal $f : R \to R$ to a DT-signal $x : I \to R$ is known as equidistant sampling. Following [22] equidistant sampling can be defined by fixing a positive real number $T > 0$, so that the DT-signal x is obtained by setting

$$x(n) := f(n \cdot T)$$

for $n \in Z$. The value $x(n)$ is called the sample taken at time $t = n \cdot T$ of the original analog signal f. This process is sometimes referred to as T-sampling. The number T is referred to as the sampling period and the inverse $Fs := \frac{1}{T}$ is called sampling rate. The sampling rate specifies the number of samples per second and is measured in Hertz (Hz).
Sampling tend to be a lossy procedure in the sense that generally information is lost in the process and, if the frequency used for sampling does not satisfy a criterion described below, the original analog signal cannot be reconstructed from its sampled representation. This constitutes a problem for many practical situations and the phenomenon takes the name of Aliasing. Temporal aliasing is then the term used to describe the distortion or artifact that occurs when a time signal reconstructed from samples differs from the original continuous signal.

Signal conversion from CT to DT is not restricted to the audio domain, but includes all aspects of signal processing. The Nyquist-Shannon Sampling Theorem is a significant outcome of the discipline. This is a necessary condition for a sufficient sampling rate that enables a discrete series of samples to capture all information from a continuous-time signal with a restricted bandwidth [22].

> If a system samples an analogue signal at a rate that is at least two times higher than the signal's highest frequency, the original analogue signal can be properly recovered from the discrete values produced by sampling.

Mathematically the sampling theorem then states that, given a CT signal f that satisfies certain conditions not discussed here, and given x be the T-sample version of f with $T := \frac{1}{2\Omega}$, then f can be reconstructed from x by:

$$f(t) = \sum_{n \in Z} x(n)\,\mathrm{sinc}\left(\frac{t - nT}{T}\right) = \sum_{n \in Z} f\left(\frac{n}{2\Omega}\right)\mathrm{sinc}(2\Omega t - n)$$

2.3.2 Quantization

While sampling turns the time domain from continuous to discrete, one also needs to replace the continuous range of possible amplitudes by a discrete range of possible values. This process is called quantization. This process is illustrated in Figure 2.1, where certain sample points do not correspond to the shape of the waveform (e.g. the third sample point) (e.g. the third sample point). Quantization introduces another undesired effect known as quantization errors, or quantization noise, which is indicated in red in Figure 2.1
Following [22] it is possible to define the process by setting a function $Q : R \to \Gamma$, referred to as the quantizer, which assign to an amplitude $a \in R$ a value $Q(a) \in \Gamma$, where $\Gamma \subset R$. In the case of digital signals, the accuracy of the amplitude values post quantization depends on the amount of bits dedicated to the encoding. For example, CD recordings employ a 16-bit encoding technique, which enables 65536 potential values to be represented.

To recapitulate, Aliasing and quantization may generate noticeable sound artifacts, such as unpleasant buzzing noises or background noise. These can however be overcome. For CD-based digital representations, however, the sampling rate and quantization precision are selected such that the deterioration of the waveform is imperceptible to the human ear. Regarding sampling rates, 8 kHz (8,000 Hz) is the common sample rate for phone lines, 32 kHz for digital radio, 44.1 kHz for CD recordings, and higher sampling frequencies up to 96 kHz are used in professional studio equipment[8] [22]. The experiments in these research deal with a sampling frequency ranging from 8kHz to 44.1 kHz and a 8 or 16 bit quantization.

2.3.3 Audio Dynamic Range Compression

The concepts of audio dynamic range compression are presented in order to give a foundation for the potential implementation of the findings of this study. First, observe that dynamic range is intended to describe the ratio between the largest and smallest signal amplitude values [9]. Then, audio dynamic range compression may be understood as a signal processing technique that reduces the volume of loud noises and raises the volume of quiet tones, thereby reducing (compressing) the dynamic range of an audio stream.
Compressors (or limiters) are the instrument that performs this signal processing technique and are comprised of four major components:

- Ratio, a parameter that regulates the amount of attenuation that the compressor will apply to the signal.
- Threshold, the amplitude level at which the compressor begins to operate. Audio signals that are quieter than the threshold are not restricted.
- Attack, the length of time provided before a signal is limited by the compressor. If the attack time is very brief, the signal will be compressed nearly immediately.
- Release, The amount of time a compressor is applied to an audio stream. When release times are extended, compression takes longer to diminish.

Another essential characteristic of compressors is the Look-Ahead. This function, as its name indicates, allows the compressor to examine a signal for a few time samples before processing it. The look-ahead feature is advantageous because it captures fast transients in the audio signals that could otherwise be overlooked. This function is computationally expensive since it requires duplicating the entire signal and delaying it by 1 to 10 milliseconds. This significantly increases the

[8]This should provide an explanation for the sensation of voices on the phone sounding ”metallic”.
[9]Not only of an audio signal, also of light and voltage signals for example

size of the audio data stream. A good ML prediction system would be able to avoid this load by relying directly on the signal as opposed to having the need of a replica.

2.4 Time Series Forecasting

Time series forecasting is the analysis of time series data using statistics and modeling to provide forecasts and guide strategic decision-making. Through the building of models, the purpose of time series analysis is frequently to gain an understanding of the sequence of events. Analysis can give insight on the "why" behind observed outcomes. Forecasting then determines what to do with this knowledge and the extrapolations of what can be predicted to occur in the future.

To formally introduce the topic, the structure of [23] is followed. To begin with, a time series is defined as a set of observation x_t each observed at a specific time, t. On the other hand, a time series model of some observed data $\{x_t\}$ is a specification of the join distributions of a sequence of random variables $\{X_t\}$ of which $\{x_t\}$ is postulated to be a realization. Most often, and also in this research, the term time series is used to mean both the data and the statistical process of which it is a realization. Nonetheless, it is essential to recognize that the assumption of the time series model is the linchpin upon which the statistical techniques used to infer the realization of the time series can be applied.

The ideas of stationarity and autocovariance function are of special relevance to the area of time series forecasting. To explain these concepts it is useful to define the mean and covariance function[23].
Let $\{X_t\}$ be a time series with $E(X_t) < \infty$. The mean function of $\{X_t\}$ is

$$\mu_X(t) = E(X_t)$$

The covariance function of $\{X_t\}$ is

$$\gamma_X(r, s) = Cov(X_r, X_s) = E[(X_r - \mu_X(r))(X_s - \mu_X(s))]$$

for all r and s.
Given such definition $\{X_t\}$ is said (weakly) stationary if:

1. $\mu_X(t)$ is independent of t
2. $\gamma_X(t+h,t)$ is independent of t for each h

Essentially, stationarity[10] indicates that the statistical features of a time series (or, more specifically, the process creating it) do not vary over time. This explain the importance of such concept, as in general most statistical and analytical tools with which time series are treated make use of this underlying assumption.
The autocovariance function (ACFV) of a stationary time series $\{X_t\}$ at lag h is defined as:

$$\gamma_X(h) = Cov(X_{t+h}, X_t)$$

While the autocorrelation function (ACF) of $\{X_t\}$ is[11]:

$$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)}$$

If there are interdependent temporal values in a time series, the autocovariance function can be used to describe this dependency. This function, as evident from its definition, compares the time series to a time-shifted version of itself. It reflects the extent to which the amplitude of a time series at one instant is connected to or may be inferred from the amplitude of a time series at a different moment. On the other hand, one may think of the autocorrelation function as a normalized autocovariance function.

Another important concept to introduce within the time series framework is that of linearity. A time series is said to be linear if it can be expressed in the form:

$$X_t = \sum_{i=-\infty}^{\infty} \psi_i Z_{t-i}$$

where $\{Z_t\}$ is a sequence of uncorrelated random variables with identical mean

[10]Throughout the paper we refer to stationarity to intend weak stationarity as in the definition here reported.

[11]Notice that sometimes $\rho_X(h)$ is also denoted as $Cor(X_{t+h}, X_t)$

and variance, i.e. a white noise process, and $\sum_i |\psi_i| < \infty$. Any process that does not satisfy the above condition is said to be nonlinear[24]. An intuition for the concept of nonlinear time series can be gained by thinking about the stock market. It is known that depending on the history period of the market, stock prices may change more or less rapidly, characterizing the market as more or less volatile[12]. This defines the presence of periods where the stock prices time series are more predictable (less volatile) depending on the history of the series. Linear models fail to capture this feature and are hence less suitable for the prediction of these processes.

2.4.1 Traditional Methods in Time Series Forecasting

Regardless of the discipline in which TSF is used, the overall strategy stays almost unchanged; The first stage is a graphical examination of the time series to determine its patterns. Specifically, we examine if the following patterns are present:

- A trend, that is a slowly changing component of the time series
- A seasonal component, that is a periodicity in the changes of the data
- The existence of outliers

This is done in order to recognize whether it is possible to undergo what is called a classical decomposition[23] of the time series:

$$X_t = m_t + s_t + Y_t$$

Where m_t is the trend component and s_t is the seasonal component and Y_t is a random noise component. This facilitates the aim of correct forecasts, as it is possible to correctly estimate and extract the trend and seasonal components, so that the task transforms in a modelling of the noise component [23].
Once the trend and seasonal components are eliminated, the noise components can be modelled. In the majority of cases, this modelling follows the assumption that the process is linear, in the sense defined in the previous subsection.

[12]Volatility is indeed a technical term within the field of TSF. The interested reader is directed to [24] and [23] for more information, particularly interesting are its presence in the ARCH and GARCH models.

This allows for the use of powerful statistical methods, the most famous of which are exponential smoothing[25] and even more the ARIMA (AutoRegressive Integrated Moving Average) models[23].

As opposed to the studies mentioned early on application of ML on audio data[9], some review studies indicates that statistical method such as ARIMA tend to provide more accurate forecasts than ML approaches[26][27]. Nonetheless, it is evident from the same experiments that NN are capable of handling difficult data, including nonlinear and nonstationary time series. In light of the fact that music audio signals are often nonlinear and nonstationary, and do not allow themselves to be classicaly decomposed, the methodology employed in this study is well justified.

2.5 Related Work

A surprising fact in itself is the lack of substantial literature on the topic, namely ML predictions of music audio data.
Audio imputation, by which it is referred the process of recovering missing pieces of audio, seems the closest aspect of research to the one investigated in this thesis [28]. Audio imputation is tangent to another kind of research that necessitates audio signal prediction for improving loss of packets in data transmission [29].

More useful has been a comprehensive study of audio signal processing provided in [9]. This reports very interesting research related to music audio signal and ML such as [30], whereby ML predictions have been used to improve automated performance of music digital instruments.
Of interest is also the Magenta project, ”An open source research project exploring the role of machine learning as a tool in the creative process”[13]. It contains ideas tangent to the problem at hand as the process of music generation resemble the process of music prediction.

Much literature is also present regarding the use of LSTM networks for time series

[13]Find it at `https://magenta.tensorflow.org/ddsp`

predictions, such as [31]. Also the review of statistical methods and ML methods for TSF is a useful study [26].

Regarding the theoretical side of audio processing, a main source has been found in [22]. This book not only well presents relevant concepts within the field of music digital analysis, but also provides examples of such concepts by means of open available jupyter notebooks[14].

[14]These can be found at www.audiolabs-erlangen.de

3 Methodology

This chapter describes the procedure that led to the examination of the particular models of this study.

3.1 Data gathering, Dataset analysis and pre-processing

Given the vast availability of music audio data and the few restrictions on the types of data investigated, data gathering was undertaken rapidly; The audio tracks used to compile the dataset were obtained from an open-source database.

All datasets have been segmented sequentially as follow: the initial 60% of audio is used for training, the subsequent 20% is used for validation and the last 20% is used for testing.
The empirical distributions of all datasets were studied in order to comprehend their statistical features. To illustrate an example, Figure 3.1 shows the empirical distribution for the toy-dataset. It is possible to notice that all the points fall within the amplitude range $[-1, 1]$ and that a clustering towards 0 is present.
In fact, for all datasets all the values of the time samples fell inside the interval [-1,1]. The sign within the range is arbitrary, the range could have been normalized between [0,1] with mean 0.5. What matters is recognizing that the amplitude represents the amplitude of the speaker vibrations, or similarly the air pressure waves caused by it. Then we see that below the mean value we would have a displacement of such in one direction and above the mean we would have a displacement in the opposite direction. Interestingly the amplitude can also be interpreted as the voltage that will cause the speaker vibrations. For this reason the raw waveform is left without physical units, and only amplitude is discussed.
The data points in both datasets exhibit what can be defined as a Normal distribution with extra mean-centered values. The latter is due to the occurrence of silence[15] in the majority of tracks.

[15]static is represented by almost still speakers and close to 0 time sample values

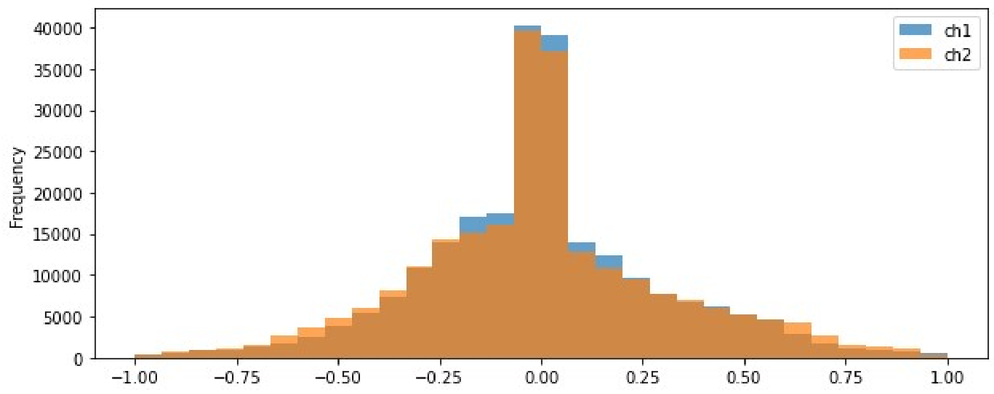
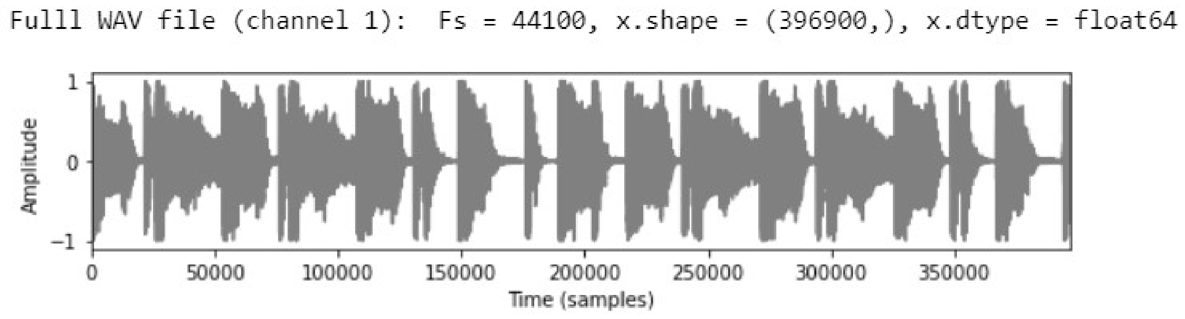


Figure 3.1: Empirical distribution of raw signal (containing two channels) for toy-data set. The horizontal axis describe the value of the time samples amplitudes. It is possible to notice the distribution closely resemble a Gaussian for the exception of a concentration of points around the 0 mean value, due to the presence of quiet sections within the audio track.

Other statistics of interest that were extracted prior to training and testing were the minimum/maximum difference between consecutive time samples, the mean difference and the mean absolute difference over all points.

While several pre-processing approaches for audio data, such as conversions to frequency domain and subsequent analysis using tools such as mel spectrograms[22], have been studied, it has been chosen to preserve the data as closely as possible to its raw form. This decision was supported by considering the algorithm efficacy on raw data as a benefit for the final application.
The only change made to the data following import was the deletion of a few outliers time samples outside the $[-1, 1]$ range. This step was deemed unimportant to the outcome of the analysis since it reduced the dataset by less than 0.001% . However, it was not possible to conclude what caused the presence of these outliers and it was therefore attributed to computation errors of the loading process.
Of importance is the consideration of the re-sampling process embedded within the loading process. The librosa.load function has been utilized with the default re-sampling process, which adopts the use of a Kaiser window. This advanced approach was not covered in the theoretical portion, but further information is



```
Fulll WAV file (channel 1):  Fs = 44100, x.shape = (396900,), x.dtype = float64
```

Figure 3.2: Raw waveform shape (entire dataset) of the toy-dataset sampled at 44.1 kHz

available[16]. However, the described sampling theorem enables us to comprehend that the audio file's frequency content will be decimated to half the required sampling frequency, effectively applying a low-pass filter to the data.

3.1.1 Toy-dataset (9 seconds audio clip)

The first dataset consists of nine seconds of continuous audio music that was hand-selected from a freely available track 3.2. A quick analysis of the dataset resulted in displaying a maximum difference in consecutive values of 1.3[17]. This result might be of consideration as it represents the largest necessary gap between time sample fed and result of prediction of the net.
The choice of track was also supported by the audible pattern that was displayed. This presented a repetitive beat that was deemed useful for the purpose of prediction as it allowed for little variation between the training -validation-test set partition.

3.1.2 Large dataset (20 minutes - 20 tracks audio collection)

With the intention of determining whether it would be possible to develop an algorithm suitable for a genre-agnostic algorithm, ie one that would perform similarly across different genres, the second major dataset for analysis was created by combining popular songs of various genres and instrumentation, including Rock, Hip-Hop, Soul, EDM, and others. Twenty songs were randomly trimmed,

[16]see for example the librosa documentation or this website.
[17]For example this indicates a change from 1 to -0.3 in one time step

with some containing the middle section, some the introduction, and others the conclusion, or an overlap of these sections.
There is no preprocessing of the data and no adjustment of the hyperparameters. A quick examination of the dataset revealed a maximum difference between consecutive values of 1.5. Important are also the Mean Absolute Differences for each section of the data set. The Mean Absolute Error (MAE)[18] for the training set is 0.0316, while the MAE for the validation set is 0.0342 and the MAE for the Test set is 0.0154. This allowed for assessing expected values of this specific dataset for baseline prediction methods.

3.2 Choice of framework

After initial trials with framing the problem as a regression task on simple signals[19], it has been decided to conduct the study using a time series forecasting framework.
A significant consequence of this decision is the requirement to provide a form of data windowing. This corresponds to the fact that the algorithm provides a series of forecasts based on a window of consecutive data samples. Thanks to Tensorflow's functionality, the window generator class could be repurposed for this work. This simplified feeding data to ML models (see Figure 3.3) and enabled for more efficient computations.
At this stage it has been proved useful to follow the procedure described in [32] in order to obtain a taxonomy of the time series problem at hand. This procedure and the answers are provided in appendix C.

3.3 Choice of ML architecture

Once the data were loaded and properly shaped,a model could be trained. Before training, the dataset was shuffled. Empirical testing revealed that nets perform significantly better when the data are shuffled, even when the signal contains sequential data. This was also confirmed by first tests on sinusoidal signals.
This behavior may be attributable to the fact that the temporal structure of the data

[18]Mathematical definition in section 3.4
[19]This topic is discussed more in the work section.

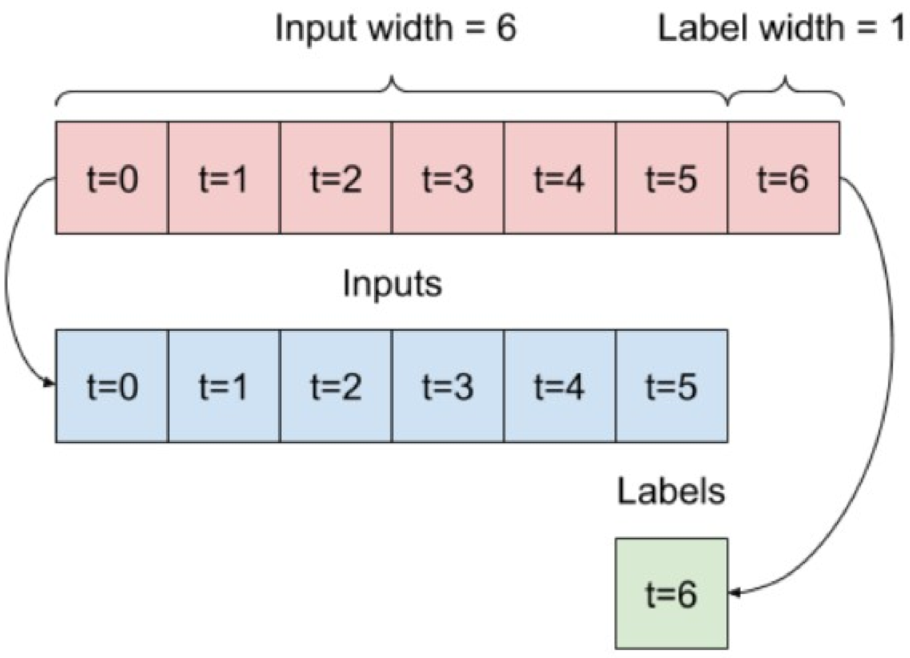


Figure 3.3: A visualization of how the window function provides inputs and targets to the model.

is enclosed within the time window and that shuffling successive windows breaks correlations between them that are detrimental to the performance of the network.

In addition to the architectural distinctions discussed in the preceding section, two choices had to be made: whether to configure the model for single-step or multi-step predictions, and whether to conduct univariate or multivariate (by using extra features relevant to the dataset) forecasts. All investigated architectures supported both univariate and multivariate, single-step and multi-step predictions.

Aside for a brief analysis on the effects of enriching the dataset with pure tones (see section 5.5 for an interesting related side result), univariate models have been preferred. Other methods of multivariate analysis, including separating different kinds of audio in different channels[20] were excluded. This was justified by the fact that, given the desired model would ultimately have the ability to generalize

[20]Here by channel it is meant a time series representation of a signal. For example using one channel for instrument type, that is using one channel for the drums, one channel for voice...

over different kinds of music audio tracks, providing an audio segmentation would bring no value.
Regarding the choice between single-step and multi-step forecasts, the latter has been preferred as better suited for the ultimate product application. Yet, in order to simplify calculations and speed up training, single-step predictions have been adopted during the analysis of other parameter influence on predictions.
After making the essential analysis design choices, the desired model architecture could be built. Keras made this process easy. The library author compares creating the design of a ML model to building LEGO block sculpture [13]. Choosing the number of layers and neurons per layer is however challenging. The results section discusses model choices.

The final step before training[21] was to select a valid optimizer. ADAM[22] [16] optimizer with default learning rate of 0.001 was mostly used. This was complemented with the choice of MSE as a loss function and MAE as a performance metric, which functions are described more in depth in the next section. Similarly the ”ReLU” (Rectified Linear Unit) activation function [23] is mostly used.

3.4 Model Evaluations and choice of metric

To evaluate the performance of models during training and testing, performance metrics are required. Notice the distinction between metrics and loss functions; Evaluation metric refers to a metric that is desired to reduce or maximize throughout the modeling process, while loss function refers to a metric that the model will effectively minimize during model training. An important difference between the two is that performance metrics need not to be differentiable[24].

For the regression part of the process the loss function has been fixed to be the

[21]or more poetically, before the possibility of unleashing the computing power of the GPU processors to identify a suitable set of model weights for the aim
[22]Adaptive Moment Estimation
[23]More information available at medium.com/@danqing/
[24]Strict requirement of the loss function for the optimization process to be valid.

Mean Squared Error (MSE), while the performance metric has been fixed to be the Mean Absolute Error (MAE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of predictions, y_i are the actual time samples values and $\hat{y}_i$ are the values predicted by the models.
MAE is computed as follows:

$$\text{MAE} = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$$

where again n is the number of predictions, y_i the actual and $\hat{y}_i$ the predicted values. Using MAE as a metric for time series forecasting is a standard procedure as it is one of the most often used metric for this task[33].
Even if MAE gives an understanding of the performance of the metric, its dependency on the dataset under consideration makes it difficult to determine the overall accuracy of the forecasts. To illustrate this, examine a comparison between model predictions on a song containing a lot of silence (maybe an indie song) a song with many loud sections (perhaps a metal song). The comparison would likely result in the former having a lower MAE. This is due to the fact that quiet portions result in low amplitudes with nearly no fluctuations, which contribute to a lesser overall MAE, while loud parts include waveforms with significant changes, which contribute to a bigger overall MAE.
To alleviate this problem, a further qualitative metric has been used.

3.4.1 The persistence forecast

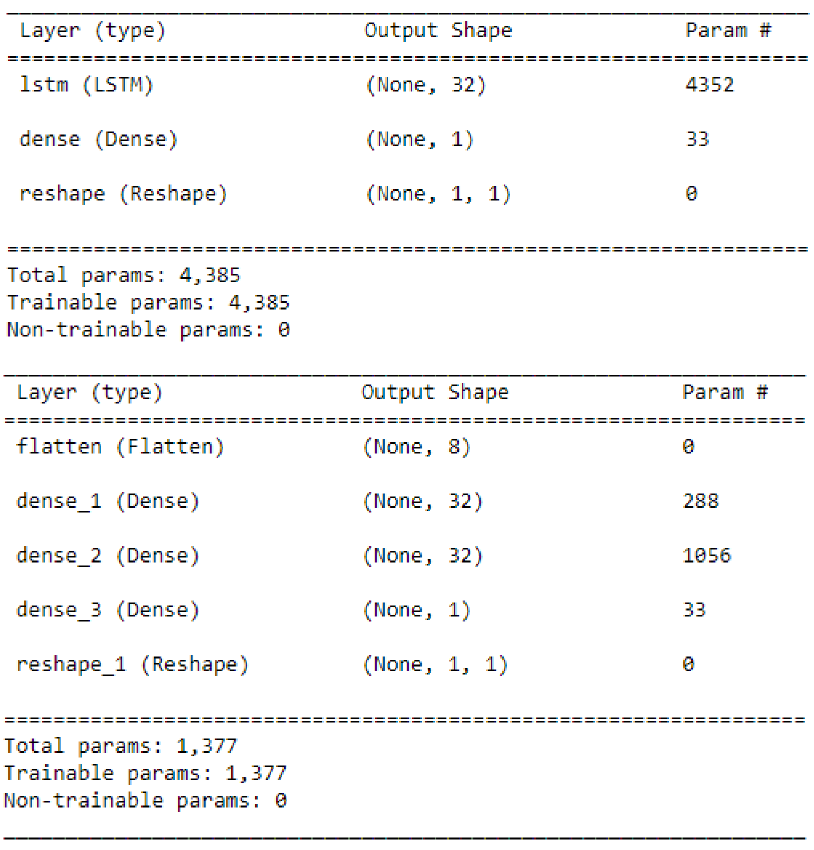
This consisted of a naïve forecasting technique in which the last time sample of a window is utilized as the regression prediction. This approach is frequently referred to as persistence forecast since the last piece of the sequence is retained as the prediction forecast.
Frequently, the persistence model serves as a guide for calculating the forecasts quality. This is because it is helpful to determine if a prediction model gives better outcomes than the persistence model, as a simple reference model [34].

The introduction of this technique made it simple to determine if a particular ML model could be considered a valid candidate. During research having a ML model MAE higher than the MAE of the persistence, was considered evidence in favor of discarding such ML model. This way, this naive forecasting strategy offered a rapid judgment tool for evaluating the model's quality. Figure 5.1a display a visualization of the persitence forecasts in case of a multiple step prediction.

3.5 The models investigated

In lieu of presenting all methods considered, this section focuses on the two top choices, the MLP and the LSTM.
A description of the model summaries can be found in Figure 3.4. It is important to notice that the RNN model is essentially composed by a single LSTM unit, followed by the dense layer necessary for prediction and the reshape layer necessary for ensuring the dataset shape. One could argue that a single LSTM unit is a valid architecture for the current task; Multiple LSTM units would have increased the likelihood of incurring in overfitting [13]. In general, simplicity is favored in this research because it is believed that a smaller number of nets parameters and layers will result to an easier look-ahead replacement implementation. Considering that a real-time application is envisioned, the latency should be extremely low. This is the rationale for preferring fewer parameters, even if it results in a performance decrease. This justifies the initial choice of 32 neurons for the LSTMs unit, which contributes to the net's restricted complexity as indicated by the total number of parameters (4385 in this particular instance with window length of 8).

Using the same broad concepts, the MLP is presented. The initial flattening layer is included to accommodate multiple input steps. Following this are two simple dense layers containing 32 neurons each. Again, it is deemed appropriate to select a small number of neurons in order to retain a small number of overall parameters. Then the net presents a dense output layer and a reshape layer which is necessary for allowing the desired choice of steps predicted.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 32)                4352

 dense (Dense)               (None, 1)                 33

 reshape (Reshape)           (None, 1, 1)              0

=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 8)                 0

 dense_1 (Dense)             (None, 32)                288

 dense_2 (Dense)             (None, 32)                1056

 dense_3 (Dense)             (None, 1)                 33

 reshape_1 (Reshape)         (None, 1, 1)              0

=================================================================
Total params: 1,377
Trainable params: 1,377
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.4: Summaries of the LSTM model (top) and MLP model (bottom)

4 Preliminary studies

In this section a description of the preliminaries studies is presented. This will include a description of the strategies taken in order to narrow down the suitable data representations and network models investigated. All significant results will be represented with a more functional sorting in the next section. As such, navigated readers are welcomed to skip directly to the next section.

4.1 First implementations

As customary, the first period involved gathering and scanning previous literature data. However, it soon became obvious that research on this specific task of music audio prediction is very limited.
The first goal of the project has been the implementation of the simplest possible algorithm which would perform audio signal prediction. A 2-case classifier has been chosen as the best candidate. Given a time sample input, the classifier task is to predict whether the next sample is of higher amplitude (Case A) or of lower-equal amplitude (Case B). To simplify the task to the most initially only simple signals, such as pure sine waves, were fed to the network.
Tensorflow and Keras libraries were used to build this 2-case classifier, while the data was shaped as numpy arrays.
The very first testing of this model resulted in a failure; the model happened to be right 50% of the cases, showing a clear lack of understanding. A posteriori it was clear this was a naïve attempt, the reason being that only a single time sample was provided to the net. This enables us to comprehend the rationale behind the necessity of giving many time samples, bundled together in a window, as well as the importance of quantifying the difference in model performance for different window sizes. In this simple case, having a window size of 2 drastically improved the performance and the predictions got to almost perfect score with a window size of 4. Once this result was reached, the performance of the net was tested on other simple signals, namely a square and a triangular wave. Again, even with short window lengths, the classifier was able to obtain perfect performance.

After these successes, the classifier performance has been tested on music audio clip. The clip has been chosen for being phonically simple and containing a repetitive beat pattern which results in a highly periodic raw waveform. This was thought to be helpful in aiding the net to good performance in predictions. The duration of the clip was of 9 seconds, representing a small dataset even when sampled at 44.1 kHz. The rest of the pipeline for creating the model remained unchanged; again the same minimal data processing was used and the model architecture remained unchanged.

Given the results on this challenge were positive but not perfect,(82% accuracy), tests were run to improve performance. An analysis of the performance of the net while varying length of inputs lead to the graph presented in Figure 5.4.
After the performance with classifiers was considered adequate, it was decided to step up the complexity by switching to regression analysis. This implied switching from predicting whether the next time sample be higher or lower than the last one, to predicting a numerical value for the amplitude of the time sample.

Much of the previous implementation was used in the design of the Regression implementation. Important steps involved avoid one-hot-encoding and switching accuracy metric to MAE, Mean Absolute Error. The investigation required the adoption of a baseline comparison to qualitatively asses the performance of the nets predictions, hence the implementation of the persistence forecast. The introduction of this strategy allowed to easily check whether any given model would improve on performance with a lower MAE compared to the MAE obtained by the naive strategy, providing a quick decision method for assessing the quality of the model.

4.2 Labelling the problem as a Time Series Forecasting task

The adoption of the persistence forecast as a qualitative metric for performance reflected a choice of central importance for the rest of the research; it was decided to frame the problem as a Time Series Forecasting (TSF) task. Even if this decision

might be taken for granted given the description of a regression task on a time sample windows, it actually represents an important distinction to many other applications of audio signal processing. In fact, most other audio related ML algorithms perform training and testing mostly on the frequency components (frequency series, spectrograms) rather than the time components of a give audio track.
The choice was made as it seemed well aligned to the scope of applications, which required minimal data preparation in order to reduce possible delays in the processing of the data. An analysis on the amplitudes of the given time samples requires the least possible amount of preprocessing, therefore a successful model obtained within the TSF framework was considered to be the ideal solution to the problem at hand.

A benefit of narrowing the problem down to a TSF task was found in the possibility of translating the common methodologies of the field to the audio prediction goal. Moreover, not only theoretical techniques but apt data pipelines were found. In the specific, TensorFlow documentation containing a TSF example[25] was studied and its methodologies adopted to conduct experiments during the rest of the project.

4.3 Implementation of different ML techniques and forecasting strategies

Once the general pipeline for experiments was tuned and optimized, different kind of ML architectures and simple forecasting strategies were tested using the same toy dataset.
Along side the persistence forecast strategy, a Linear model was tested as a possible viable candidate for bench-marking. In essence this model consisted in a linear transformation between input and output whereby the output depended on a single time step (i.e. window length is 1) and was produced by means of a model containing a single layer with a single neuron. Models so trained resulted in a worse performance compared to persistence forecast and hence were discarded.

[25]www.tensorflow.org/tutorials/structureddata/timeseries

Even when increasing the complexity of the linear model, from a single layer with a single neuron to a deep network with two hidden layers of 32 neurons each. This reconfirmed the very first result of this research, showing that not enough information is carried by a single time sample to result in a valid prediction of the following time sample. Hence also these models were discarded.

The performance quality drastically changed when the models were allowed to take multiple time steps as input to produce a single output. To do so the number of input neurons where adjusted to match the window length, which for the first trials was fixed at 16 time samples. These MLPs models resulted in a lower MAE when compared to the persistence forecast and were therefore kept for testing on different parameters investigations.
The best performing architecture of those tested turned out to be Recurrent Neural Networks. These were implemented by means of a single LSTM layer composed of 32 units. Unsurprisingly the performance is of higher accuracy when the return sequence is set to false and this process was used during the major part of parameter analysis.
The next phase of research involved a series of analysis on the dataset properties and the effect of changing parameters and model hyperparameters on prediction accuracy.

4.4 Increasing Dataset size

Once an overview of the different model performance was obtained using the small 9 second audio clip, the dataset was increased. First some of the analysis were replicated on a 6 minutes audio track and later on a clip of 20 minutes duration containing 20 one minute segments from songs of different genre.

5 Results

This section presents the key findings of the research. These will be discussed according to their classification by type of analysis and utilised dataset type. In appendix A.1, it is possible to review significant code fragments.

Based on modifications in window length, sampling frequency, and number of prediction steps, the research examines the accuracy of predictions. Here lower MAE are desirable. Understanding the effect of these parameters on net performance and determining the optimal range for each is the objective. This type of evaluation is vital when determining the suitability of a machine learning algorithm.

Figure 5.1a gives a summary of the kind of studies undertaken. Given as input a variable window length consisting of time sample amplitudes that represent the raw waveform of a given music audio clip, the objective of the networks is to correctly estimate the subsequent time sample amplitudes by matching them as closely as possible to the actual values of the track. The same picture shows the persistence forecast's type of output which the networks are compared against. This is displayed in the case of a 8 step predictions in Figure 5.1b.

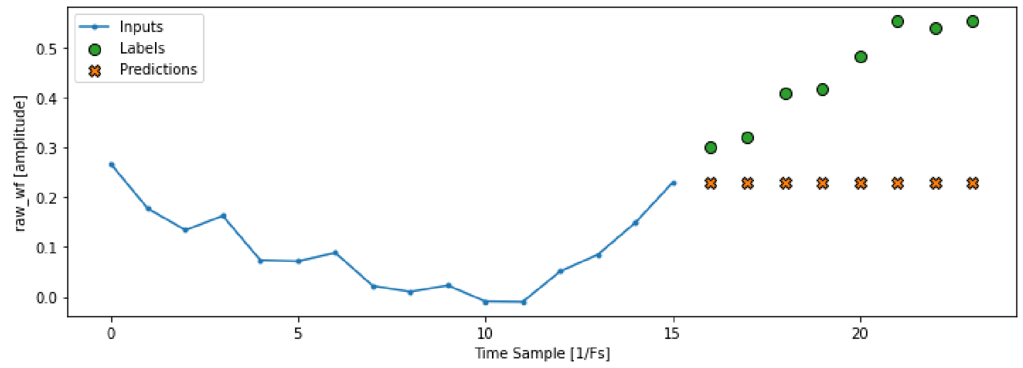
5.1 Analysis on window length

The analyses were originally performed on a small toy-dataset before being replicated on a bigger dataset.

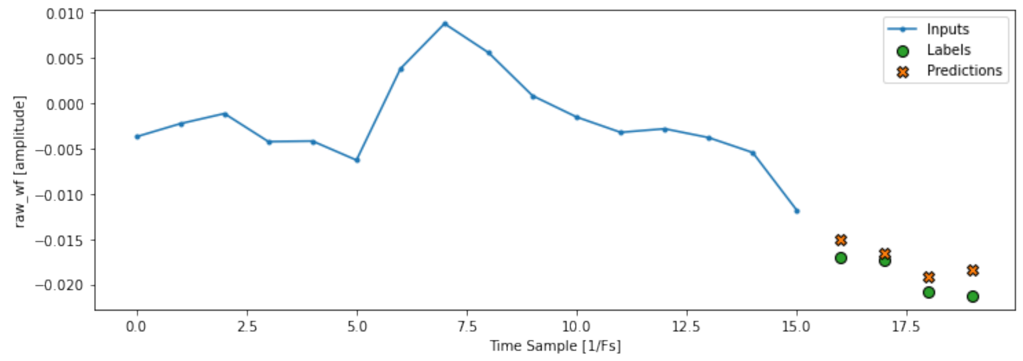
5.1.1 Toy-dataset

To focus on the effect of varying the window length, a single prediction step analysis was chosen. Similarly, the Sampling Frequency was kept fixed at 22050 Hz, resulting in a dataset size of 178605 double float time samples. The Window Length varied from 2^3 to 2^8 in integer steps of exponent.

The results are presented graphically by first displaying the results of the single LSTM compared with the persistence forecast 5.2a, the analogous results for the



(a) Visualization of Persistence Forecast on 8 prediction steps.

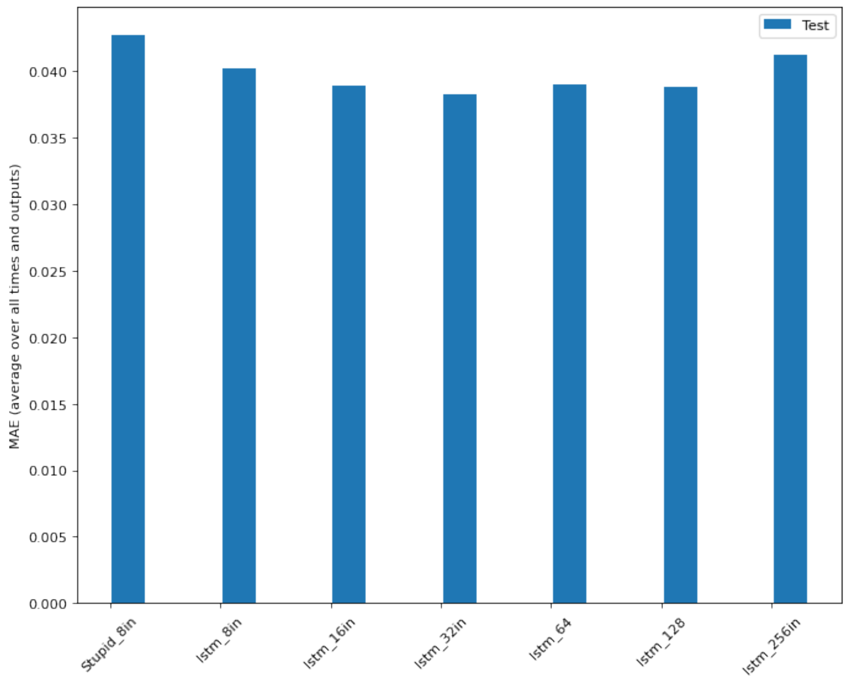


(b) Visualization of LSTMs prediction on 4 prediction steps.

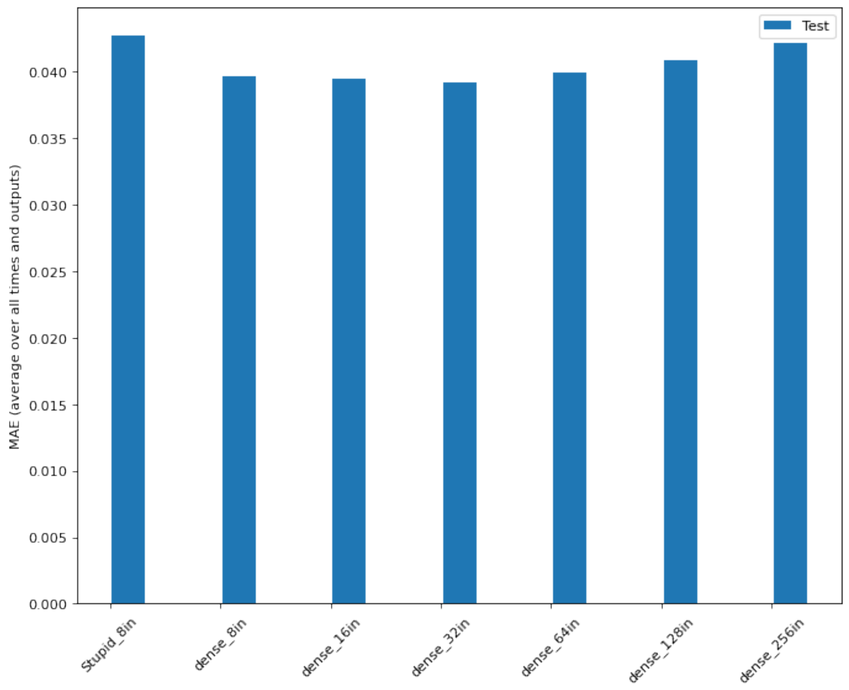
Figure 5.1: Example of the network functioning taking an input size of 16 sample (a variable window length, this is shown by the segmented blue line) and producing a given n step prediction (marked by orange crosses) and the corresponding correct labels used for training (marked by green dots)

MLP 5.2b and finally a comparison of the two 5.3.

Observing Figures 5.2a and 5.2b, an intriguing pattern is discernible that applies to both MLPs and LSTMs. As expected, it appears that the net's performance initially grew as the window length was raised, but after reaching a critical value, in this case 32 inputs, the net's performance began to fall continuously. This peculiar occurrence could be attributed to insufficient training for the NNs with more than 32 input units, but the same pattern was observed when the analysis was repeated without an early stopping criterion and with a longer training period (20 epochs) for networks with a greater number of inputs. It is unclear whether even longer training time would reconfirm such behaviour.
This phenomenon has already been encountered in initial testing on classifiers. We display the similar behaviour in Figure 5.6.Although it is evident that the accuracy of the net decreases as the window size increases proportionally with



(a) Window Length analysis on toy-dataset: LSTM prediction accuracy on test set for varying window length



(b) Window Length analysis on toy-dataset: MLP prediction accuracy on test set for varying input number

Figure 5.2: Window Length analysis on toy-dataset: Results of MLP and LSTM predictions on test set on varying of input number

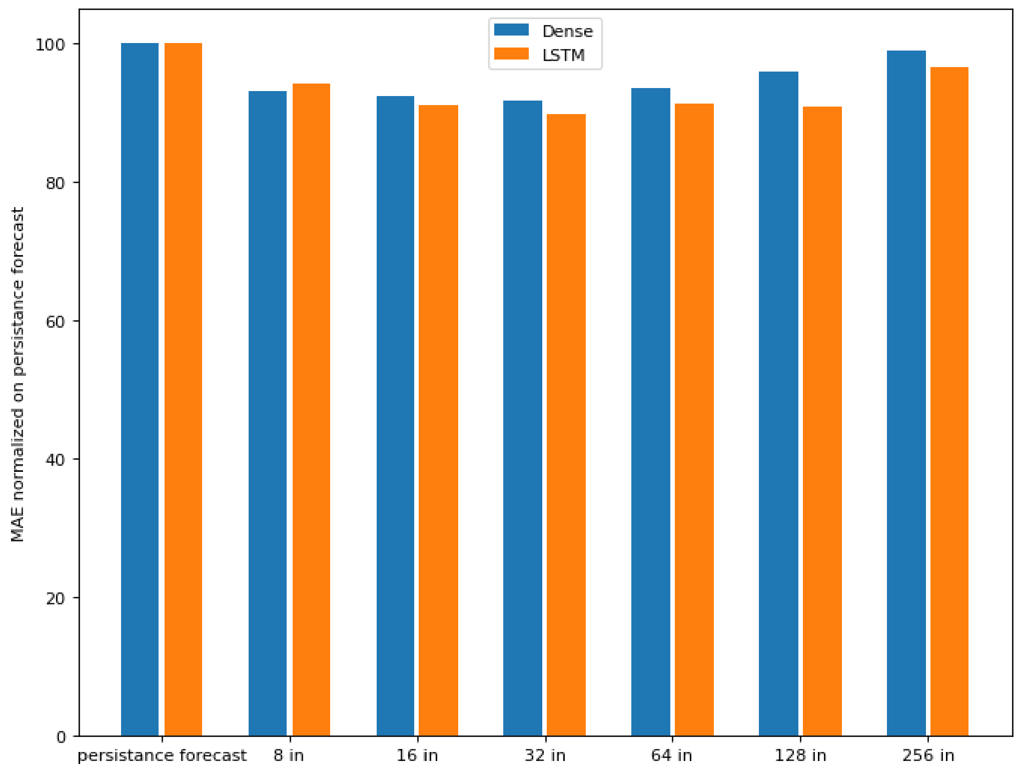


Figure 5.3: Window Length analysis on toy-dataset: Comparison of MLP and LSTM predictions normalized on the value of persistence forecast output

the amount of inputs, the explanation for this tendency is uncertain. A plausible explanation is that the finding is an artifact resulting from the difference in amount of training of the nets. Given that a greater number of inputs meant a greater number of parameters, the training requirements for the large window NN may not be met. Consequently, the graph may represent this lack of training. However, this theory is not supported by the repeated occurrence of Figure 5.2a's behavior despite unequal training time of the nets.

Then it is conceivable to assert that such a result is due to a decline in the importance of data points farther from the prediction's most recent time sample (i.e., time sample with lag 1).This can be explained as follows: as the number of inputs, and consequently the number of samples with a large lag time, increases, the short lag time samples are given a lower weight influence on the net's prediction. Coupling this fact with Figure 5.5, which shows that samples with shortest lags have the largest autocorrelation coefficient amplitudes, it is reasonable that the total outcome is a poorer forecast.

The preceding assertion has not yet been established, as it is unclear whether the autocorrelation coefficient is a suitable metric for the overall information transfer. Nonetheless, if the assertion is correct, this would be favorable for the research's ultimate objective, as it would indicate that small window lengths are advised for optimal prediction performance.

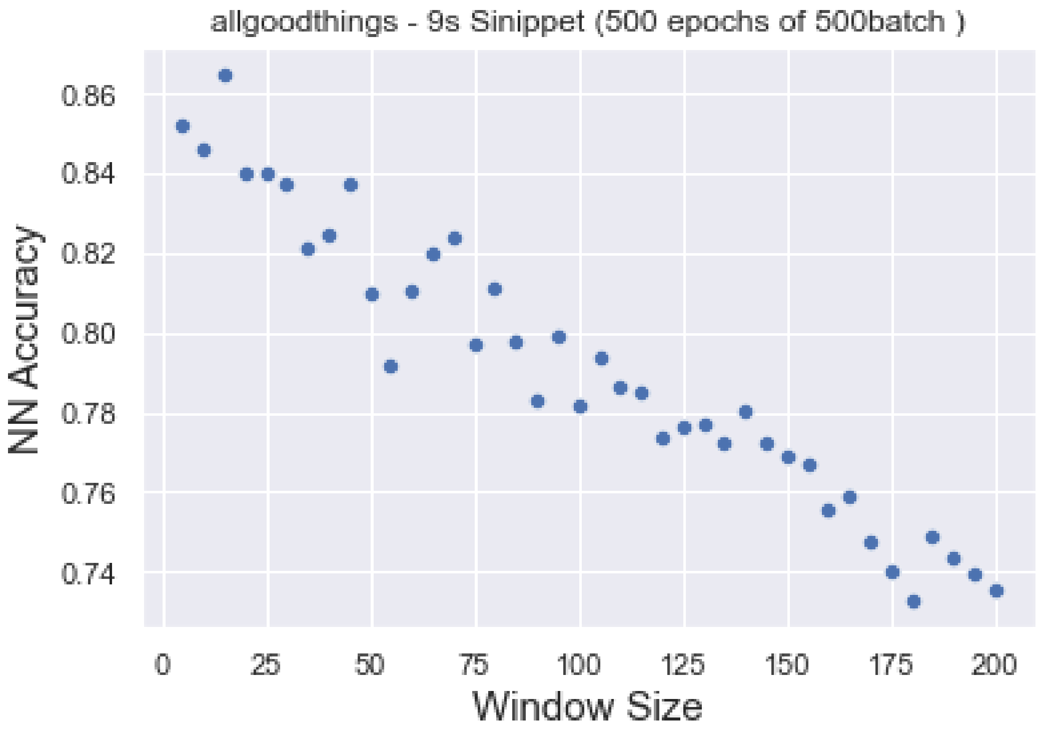


Figure 5.4: 2-case Classifier accuracy vs Window Length on toy dataset. Observe that the prediction accuracy decreases with increasing window length

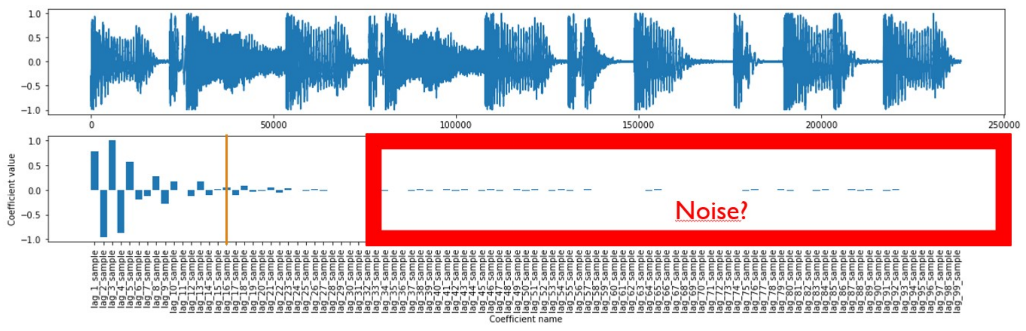


Figure 5.5: Raw training dataset waveform Sampled at 22050 Hz(top) and value of Auto-Regression coefficients for the first 100 lag values after the 32nd sample(bottom). Notice how the amplitude of these coefficients decreases as the lag increases, indicating lower correlation between the given sample and its lagged companion. With the orange line it is marked the 16th time sample, and the red box marks all samples past the 32nd.

Last but not least, figure 5.3 provides two additional insights: First LSTMs and MLPs outperform the qualitative metric for all investigated window lengths indicating a positive outcome towards the final goal. Second, LSTMs tend to outperform MLPs indicating they could be better candidate for ultimate implementations.

5.1.2 Large dataset

The analysis performed is similar to the one previously described, with the exception of a change in the dataset and a reduction in window lengths to reduce computational costs.

Also for this experiment, only a single time sample was predicted as output, and the sampling frequency was fixed at 22050 Hz. This resulted in a total of 1,867,635 time samples. It was decided to keep a relatively high sampling frequency in light of the results depicted in Figure 5.7, wherein a greater sampling frequency resulted in a significant improvement in the accuracy of prediction.

Figure 5.6 depicts the outcomes for window lengths of 16, 24, and 32. Here, little variety in the outcomes is observed, and there is no discernible pattern of decreasing predictions with increasing window size. This is attributable to the small range of inputs tested, and it is hypothesized that similar behavior would be observed with larger window sizes. There is however confirmation that the nets continue to beat the Persistence Forecast, albeit with a slight improvement. This could correspond in a higher difficulty of the nets to learn abstract patterns to apply in the case of such a diverse dataset content.

To remedy the limited window length range of the above study, LSTMs were tested also on very large window lengths. The same dataset was used, but only 90% of the dataset was used for training and testing, resulting in different values for MAE of the various sets compared to the one displayed above. Of interest was the amount of training time required for performing such a graph, which required around 10 hours of processing for a window length of 100 and more than 24 hours for a window length of 400.

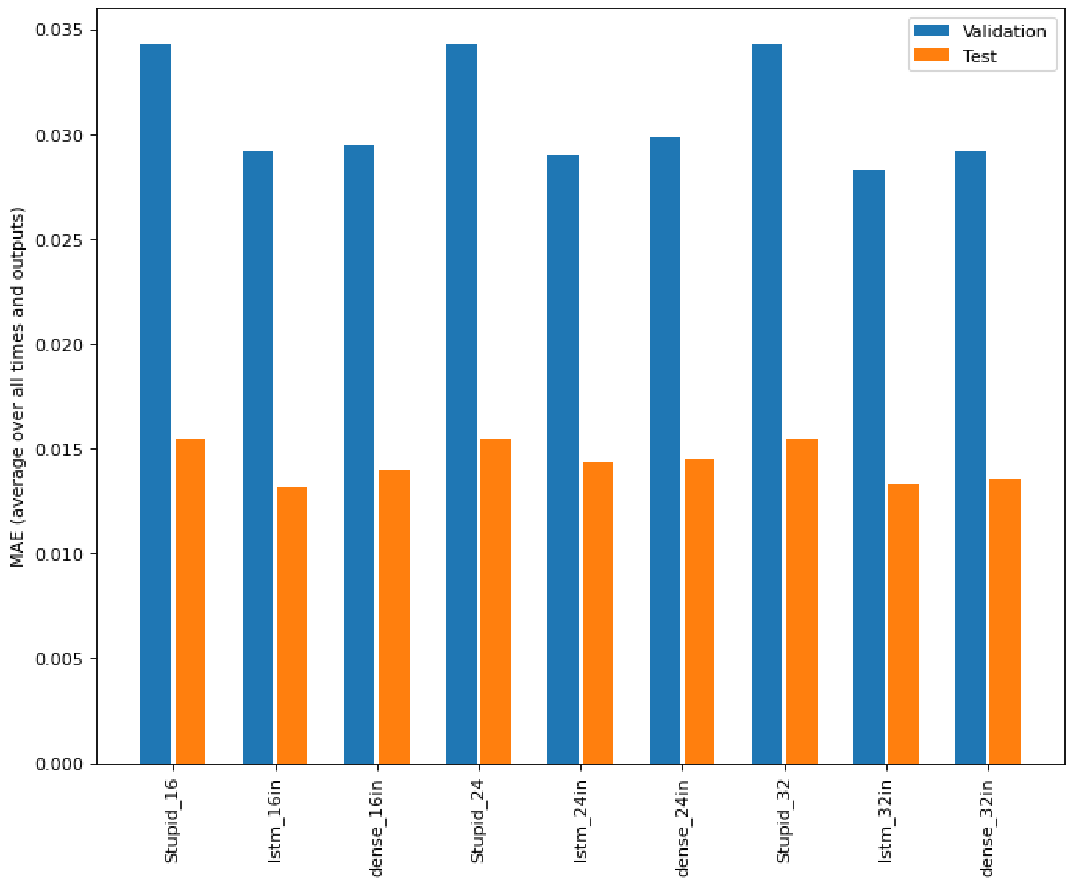


Figure 5.6: Window Length analysis on large dataset: Prediction values of validation and test sets of both LSTMs and MLPs(dense) models versus the Persistence Forecast(Stupid).

5.2 Analysis on sampling frequencies

In order to optimize for computing efficiency, the window length was set to 16 and the prediction steps were set to 1, therefore forecasting a single time step.
The sampling frequency varied from $\frac{44.1}{1}$ kHz to $\frac{44.1}{7}$ kHz in integer steps of the divisor.

5.2.1 Toy-dataset

The results are described by direct analysis of Figure 5.7. Observe that the first, fourth, seventh, etc. items in the histogram are consistently greater than the subsequent two. This result illustrates the useful discovery that both MLPs and LSTMs outperform the Persistence Forecast at all evaluated sampling frequencies. Reducing sampling frequencies correlates with an increase in the MAE for both

the test set and the validation set. This behavior is anticipated given that a lower sampling frequency equates to a coarser partition of the same waveform and, consequently, a bigger average difference between successive time samples.
Notably, the disparity between net performance and qualitative metric is significantly more evident for the 44.1 kHz SF compared to the other sampling frequencies, particularly in the LSTM example, where the performance is more than twice as good as the Persistence Forecast. This suggests that high sample frequencies may be necessary for accurate net forecasts.

The amount of training time allocated to each network is not apparent on the graph but should be noted. Even while it may appear that MLPs and LSTMs generate comparable performance at lower sampling frequencies, this is partially attributable to the early stopping requirement, which causes LSTMs to consistently terminate training before reaching the maximum training time of 10 epochs. In contrast, the majority of MLPs reached their maximal training duration.

5.2.2 Large dataset

Given that the results exhibited in Figure 5.8 closely resemble the behavior of the results stated for Figure 5.7, one could claim that scaling the dataset size has no effect on the rationale presented in the previous section. Hence this might indicate that suggesting a larger datasets will not affect the performance of the model on the sampling frequencies analysis. Also, the same consideration of LSTMs stopping training prematurely (after around 6-7 epochs) remains valid in this instance.
This dataset's analysis does not contain the 44.1 kHz frequency since the dataset was too large to be processed on a personal computer. Therefore, it is impossible to determine whether a significant improvement in predictions due to the increased frequency would also be present in the large dataset. It is however possible to observe a consistent but slight rise in MAE for forecasts at lower sampling frequencies.

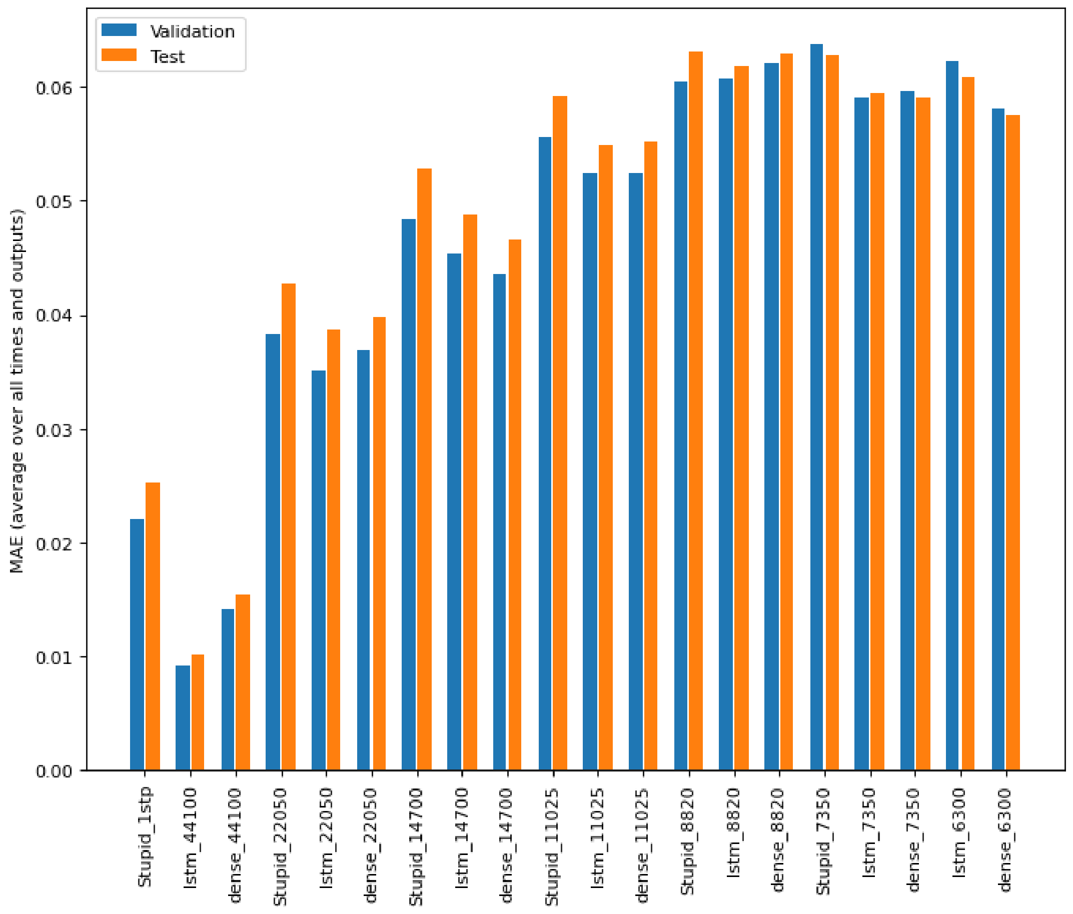


Figure 5.7: Sampling Frequency Analysis on toy-dataset: Combined results on validation and test sets of LSTMs and MLPs (dens) model versus the Persistence Forecast(Stupid).

5.3 Analysis on number of prediction steps

The sampling frequency was fixed at 22050 Hz, and the window length was set to 16. Due to the previous results, only LSTM nets were investigated for this task. The accuracy of the LSTM was evaluated by altering the output lengths from 2 to 16. Similar to the previous section, increasing the size of the dataset does not appear to affect the study's conclusions. Consequently, only the results of large dataset will be provided.

5.3.1 Large dataset

It is noteworthy that, surprisingly, almost no LSTMs had an early stopping in training during this experiment. This may be a result of the bigger error values caused by an increase in the number of forecasts, which lengthens the time

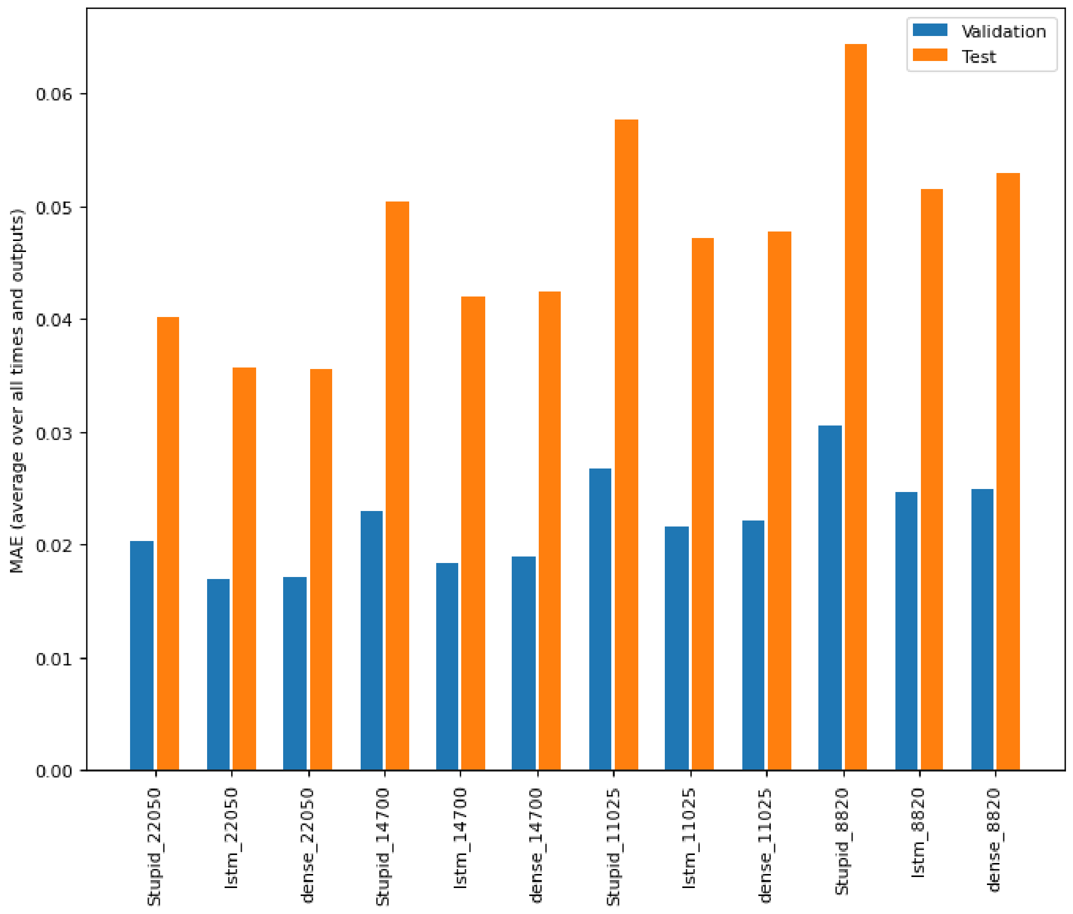


Figure 5.8: Sampling Frequency Analysis on large dataset: Combined results on validation and test sets of LSTMs and MLPs (dense) model versus the Persistence Forecast(Stupid) on larger dataset.

required to locate a local minimum or due to the higher variation n gradients cause by the larger dataset.

As can be seen in Figure 5.9, a larger number of prediction steps results in a higher MAE for both test and validation predictions. In both cases, however, LSTMs beat Persistence Forecast, suggesting that confidence may be maintained regarding the optimization of an algorithm for forecasting several future time steps.

Examining Figure 5.10 will provide a more intuitive perspective. Here, one can see Figure 5.9's test results normalized by the persistence forecast for 2-step nets. Using such outcomes as a starting point permits inferences regarding the forecasts. Firstly, it is possible to observe that the performance of LSTMs is around 10 % better than that of the respective persistence forecasts. In addition, it is possible to observe that the MAE seems to increase practically linearly as the number of prediction steps doubles.

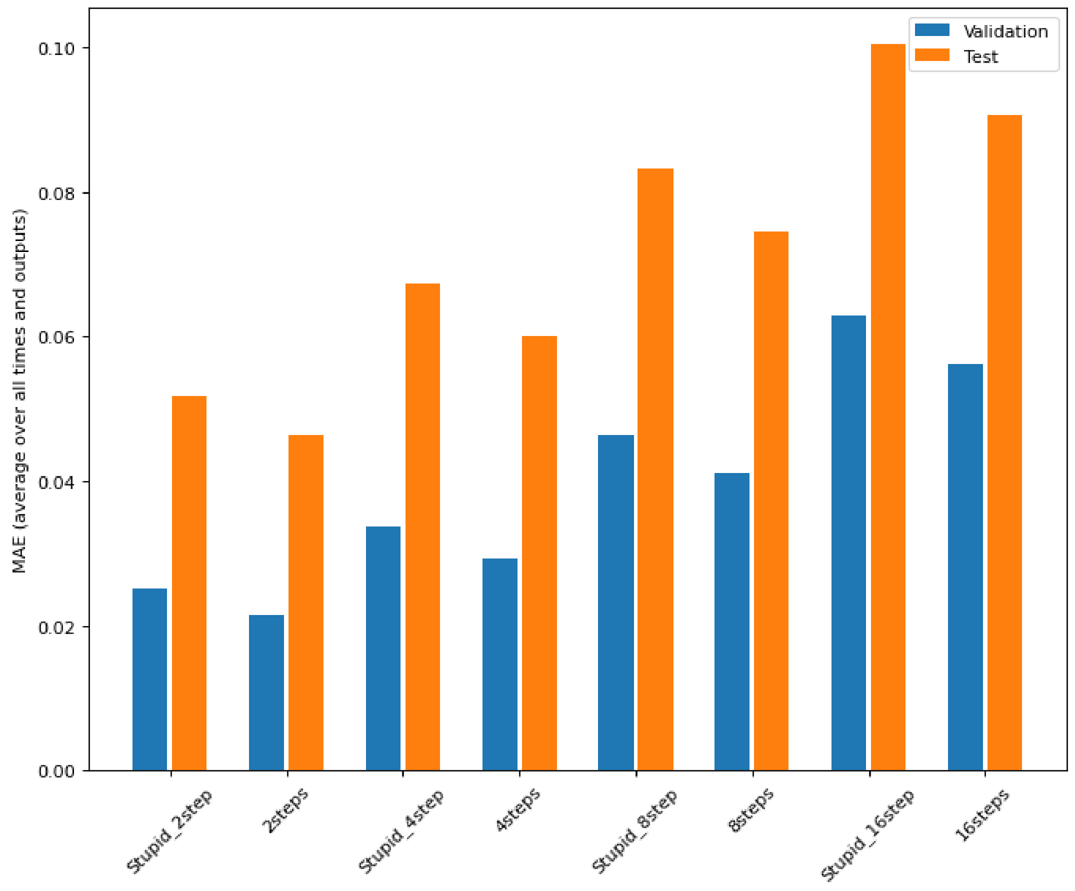


Figure 5.9: Prediction steps analysis on large dataset: Results of LSTMs accuracy (nSteps) compared to Persistance Forecast (Stuipd nSteps) for values ranging from 2 to 16 predicted samples.

5.4 Varying Sampling Frequencies while maintaining Window Length and Prediction Steps fixed in time (Look-ahead input replica analysis)

In order to better comprehend the potential performance of a NN as a look-ahead replacement, this investigation examines the accuracy of the nets in simulating the input-output requirements of a conventional look-ahead.

In this instance, the sample frequencies are varied similarly to the previous analysis, but the window length is modified so that about 10 milliseconds of audio

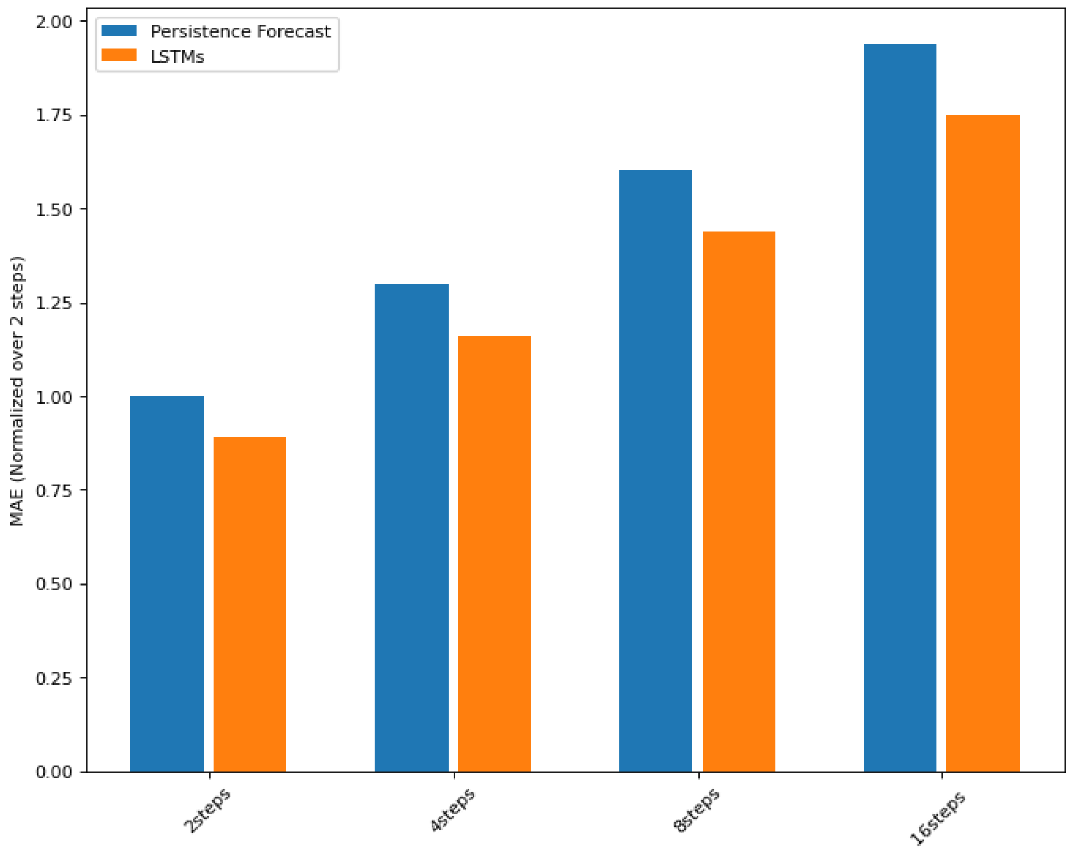


Figure 5.10: Prediction steps analysis on large dataset: the test set results presented in figure 5.9 are here reshaped with a normalization based on the 2-steps persistence forecast.

Table 5.1: Parameters setting for replicating traditional look-ahead input. Notice window length corresponds to 10 ms of audio time and the steps predicted form 1 ms of audio time

Sampling Frequency	Window Length	Steps Predicted
22 kHz	220	22
14.7 kHz	147	15
11 kHz	110	11
5.5 kHz	55	5

are received and the prediction steps are modified so that 1 millisecond of audio is predicted, see Table 5.1 for clarifications.

Figure 5.11 display the results of this analysis and Table 5.2 presents the same results in tabular format.

This investigation yields a number of noteworthy findings. To begin with, we observe that the Persistence Forecasts is substantially equal for all sample frequencies and related windows. This demonstrates that, given fixed input and output in time, a change in successive time sample differences does not result in a change in the overall mean difference, regardless of the signal's coarseness or fineness.
The more striking discovery is that the identical behavior is observed in LSTM predictions. Indeed, regardless of sampling frequency, the MAE values of the predictions cluster densely around 0.087. This may suggest that the most important factor contributing to MAE value is actually the amount of time, as compared to the amount of time samples, contained in a window.

In this instance, MLPs clearly outperform LSTMs, which is an additional interesting finding. It must be mentioned that the decision was made to rearrange the MLP's hidden layers from the architecture previously displayed. In this instance, there is a single hidden layer with 512 neurons. This is coupled to an output layer with a neuron count dependent on the prediction step selection. In the case of a 22kHz sampling frequency, for instance, the hidden layer is coupled to an output layer of 22 neurons, resulting in a grand total of 12,310 trainable parameters. For large datasets, the number of net parameters appears to be the determining factor for net performance. In fact, the MLP tested on 220 inputs and 22 outputs is the most effective algorithm. Comparisons with the LSTMs network, which received the same amount of training time but has fewer than half of the trainable parameters (5078), and MLPs trained on other sampling frequencies and with few output neurons corroborate the previous statement.

Table 5.2: Look-ahead input replica analysis on large dataset: MAE for the test set of persistence forecast, LSTM, and MLP based on the selected Sampling Frequency. Observe that the sample frequency value determines input and output for the NN architectures shown in Table.5.1

Samp. Freq.	Persistence	LSTM	MLP
22 kHz	0.1112	0.0877	0.0404
14.7 kHz	0.1123	0.0874	0.0503
11 kHz	0.1111	0.0859	0.0580
5.5 kHz	na	0.0807	0.0733

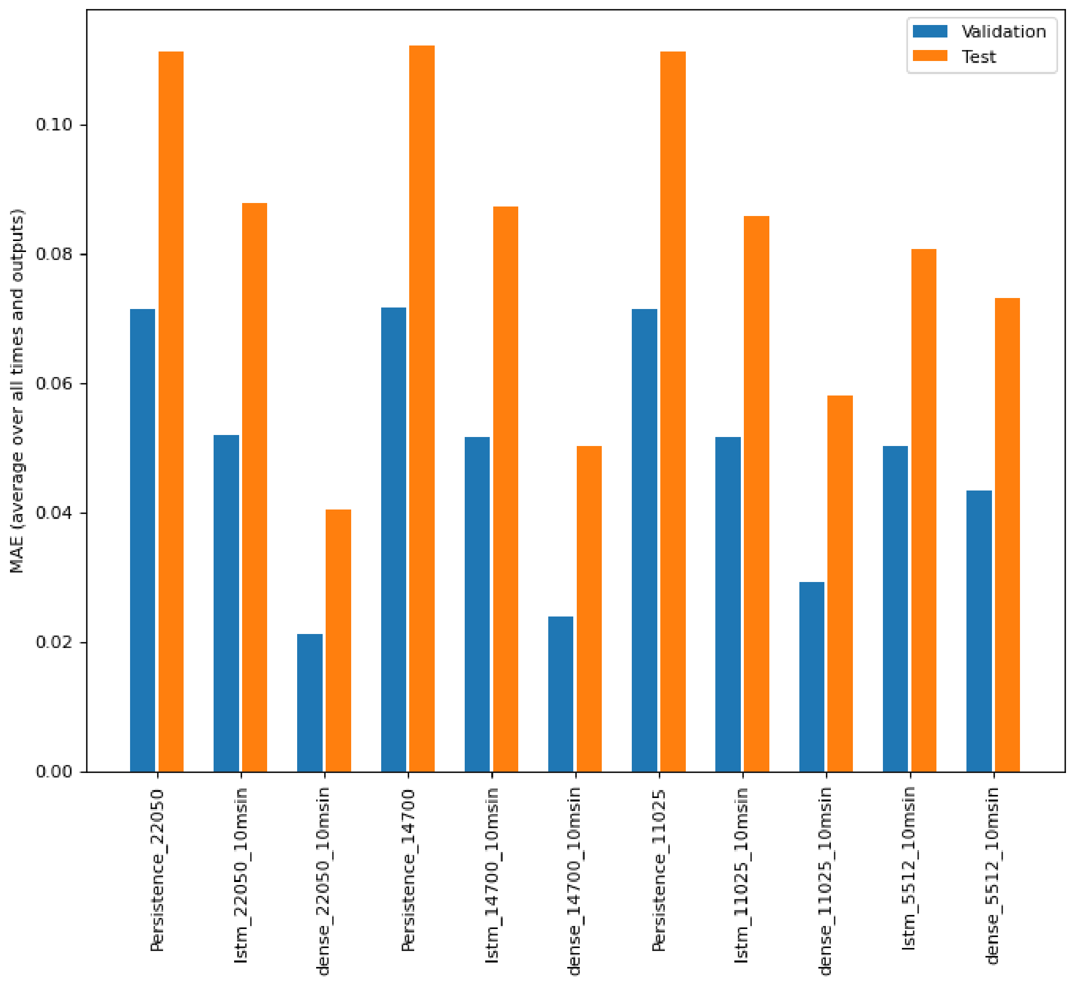


Figure 5.11: Look-ahead input replica analysis on large dataset: The histogram show the results in prediction of the MLPs and LSTM for different Sampling Frequencies

5.5 Side result - A perspective on complexity and scale

This part aims to demonstrate an interesting effect that was discovered when evaluating the multivariate LSTM model's predictions after training on the toy-dataset. Here, the additional features in training that compose the multivariate predictions correspond to enriching the model by adding pure tones (sine waves) pertinent to the signal spectrum to the raw audio files. Four frequency components, 50Hz, 500Hz, 1000Hz, and 20000Hz, were incorporated.

The effect is manifested while visualizing and hearing the results of the net's prediction on the added pure frequency tones. Figure 5.12 illustrates the

predictions of the LSTM on the 500Hz signal. Judging exclusively from the top part of Figure 5.12[26] one might assume that the net's predictions are close to perfection. Examining closely the bottom part of the figure, which display almost exactly a single period of the net's pure tone replica by scaling down the top part by a factor of 10, it is possible to notice some variations from a pure tone (indicative is the roughness around time sample 20 and 70).
Now the interesting result; scaling up the number of time sample to the 100,000 range and playing the resulting signal on loud speaker results in realizing that the deviations from the pure tone actually carry information related to the raw audio file comprising the dataset. More explicitly, the beat of the audio file tracks "leaks through" the neural net weights into its prediction of the pure tones.
While this result might be considered as a pure undesired effect for the final goal of the research, the author finds a fascination in this display of emergent behaviour due to the change of scale in time. While by simple visual inspection the signal appears devoid of meaningful information on the 10 milliseconds scale, and appears containing random noise on the 1 millisecond scale, scaling up to the seconds and playing the signal to loud speakers allows to understand the significance of such tiny deviations, giving birth to rhythm underneath the pure tone sound.

This is of help for the later section, as it explains the apparent better accuracy of test predictions in comparison to validation predictions, as the MAE within the test was initially smaller.

[26]Here are displayed 1000 time samples from a segment of the dataset with a sampling frequency of 44.1 kHz. This corresponds approximately to 0.02 seconds of audio and given the frequency of the sine is 500 Hz the figure displays approximately 10 cycles

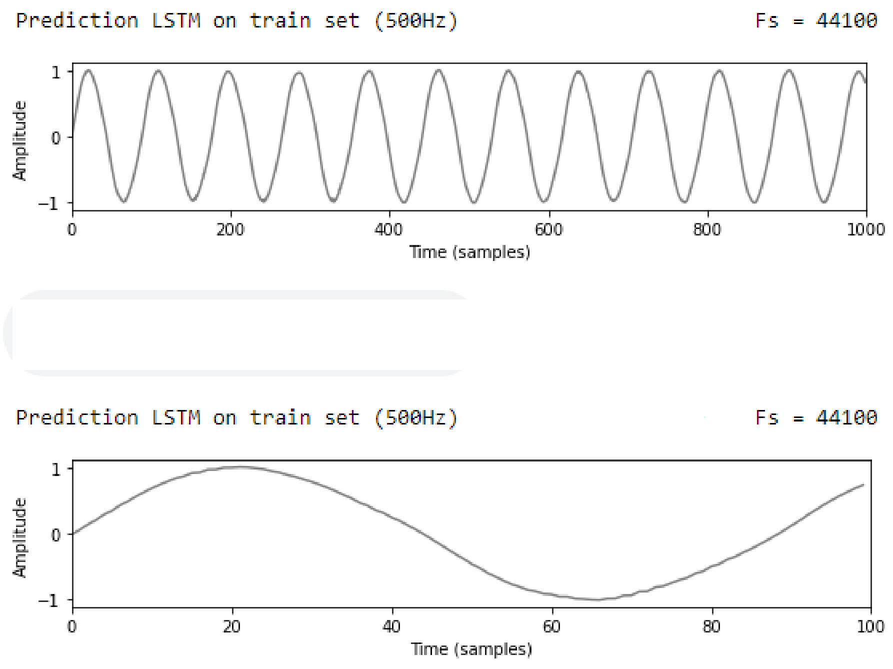


Figure 5.12: Multivariate LSTM model output prediction for 500 Hz signal of enriched toy-dataset. On the top we see net's predictions over a 1000 time sample spans, while on bottom we see approximately one single cycle spanning 100 time samples.

6 Conclusions

This section is dedicated to reflect upon the information provided by the study and discuss ways forward.

6.1 Best ML model

From the results of the research it is not possible to state that the investigated neural networks are the best candidates for the requested algorithm. Following a via negativa, it is however possible to exclude models that are proven to be suboptimal.
Between the models to be excluded it is possible to find all classification based methods, as the successful implementation of a look-ahead replacement would request too many classes for proper performance.
Within the regression methods, it is maintained that analysis within the temporal structure is the correct approach for the task.

Coming to the specific architectures, linear dense networks and any other dense network which do not take into consideration a history of time samples (i.e. all values within the window provided) are effectively too simple to be good performers. These have been found to perform worse than the persistence forecasts in a number of cases. Moreover also autoregressive models, whereby the output of the nets are utilized as data for the subsequent forecast, such an autoregressive LSTM should be considered inadequate for the task at hand. This is because predictions of multiple time steps tend to deteriorate in quality with autoregressive models and the use of the autoregressive forecasts are not a necessity given the data type[27]. Hence a proper adjustment of window length can effectively lead to performance improvement over the use of autoregressive models.

As seen in Table 5.2, MLPs are deemed the most effective by numerical testing. However, this only became apparent late in the study process, and owing to time

[27]As stated in precedence, any type of music audio will automatically have attached the labels required for supervised learning.

restrictions, it was not possible to confirm this via more tests.
Given the same number of net parameters, it appears plausible to assert that LSTMs are superior to MLPs. Nonetheless, the latter have the benefit of a relatively straightforward architecture and faster training than LSTMs, even when the number of parameters is increased. In this study, both designs are therefore regarded to be legitimate choices. In order to corroborate this assertion, more complex architectures, such as WaveNets[35], a deep neural network for generating raw audio and used in the Magenta project, need be researched.

6.2 Window length analyses

Probably the most intriguing finding of this study is that an increase in the window size does not necessarily lead to an improvement in the accuracy of forecasts. Following the description provided in the result section, it has been hypothesized that this is because a longer window length causes the nets to give too much weight to highly lagged time samples and too little weight to short-lagged time samples. This argument is not however protected from criticism. In fact, the explanation provided in the result section is not granted to hold due to the time series not being weakly stationary. As such it could be argued that a suitable network should be able to disregard the irrelevant information. This would then imply that the results is an artifact resulting from insufficient training of the larger, higher number of inputs, networks.

6.3 Sampling Frequency analyses

Downsampling is a valuable strategy for accelerating training. It emerges from the research that it is not necessary to retain the sample frequency at its maximum, but rather reducing it yields sufficiently correct predictions. Given the correlation within the downsampling and the decimation of frequency contents, this result might also suggest that there are no frequency bands which are easier to predict compared to others.

6.4 Number of outputs predictions analyses

Given the results of section 5.4 and 5.5, it is possible to conclude that suitable multi-steps forecasts are possible. If by one hand it encouraging the fact that the MAE error tends to grow linearly with the doubling number of prediction steps, on the other it is crucial to note that this fact has only been tested on a small number of outputs. Therefore it is anticipated that this fact will not hold true for a large number of outputs. Anything greater than one-tenth of the sampling frequency should be regarded as a significant number of outputs. The number of outputs is then encouraged to remain below this level and ideally approximately one-tenth of the number of inputs. These results are nevertheless regarded capable of satisfying the look-ahead replacement criteria.

6.5 Discussion

The results imply that adopting a NN as a look-ahead is feasible and could produce positive outcomes. For example, there would be no need to duplicate and delay a track (a common application of a look-ahead); rather, the net might utilize the same data stream as the compressor and decide whether to activate the compressor based on the net's projection. In fact, even if these predictions were not totally accurate, the basic operation of the compressor would still be ensured so long as the error in predictions was minimal. If the forecast error is too big, there is a potential that the compressor will miss a quick transient or run in undesired or unnecessary sections of the track, which could be a substantial disadvantage. All of the preceding arguments should confirm that it is possible to feed a NN raw music audio data and obtain a valid forecast.

It is encouraging to note that the research revealed a strong link between the results and the planned implementation objectives; the window length analyses revealed that a short window length and, consequently, a low computational goal may be best for making accurate forecasts. In certain situations, higher sample frequencies produced the better results, but the analysis of sampling frequencies revealed that downsampling can be done successfully while preserving accurate forecasts. The multi-step analysis demonstrated that a relatively large number of prediction steps do not invalidate the quality of predictions. All of these outcomes are positive. However, the lack of genuine comparisons with more sophisticated results is a significant negative, as the studies should be performed also on such techniques to verify its validity. Hence it was not possible to establish which supervised learning method is best. One may add that the window length assessments do not provide sufficient rigor to conclude that the result presented is ironclad. A similar lack of rigor prevented the determination of an optimal number of layers and number of neurons per layer.

Overall, the results indicate that the original concept is practical and worthy of additional effort.

6.6 Future Work

It is suggested that this research be advanced by beginning with the results presented. The first suggestion for future research would be to compare the analysis presented here with more complex approaches, using the rather small data set employed in this study. Once it is determined whether simple methods are more effective than complex ones, it is suggested that additional comparisons be conducted to determine whether the nets can generalize across a variety of musical genres or whether they should be trained for specific musical genres or musical instrument content. It is suggested that this type of inquiry utilize sufficient processing power, such as a computer cluster.

References

[1] "Mittheilung, einen von König Oscar II gestifteten mathematischen Preis betreffend- URL -". In: *Acta Mathematica* (1885). DOI: 10 . 1007 / BF02402191.

[2] "problemè de trois corps et les èquations de les dynamique - URL -". In: *Acta Mathematica* (1889).

[3] Newquist, HP. *The Brain Makers*. NY: The Relayer Group, 2017, p. 491.

[4] Hershey, Shawn et al. "CNN Architectures for Large-Scale Audio Classification". In: *CoRR* abs/1609.09430 (2016). arXiv: 1609.09430. URL: http://arxiv.org/abs/1609.09430.

[5] Wang, Yuxuan et al. "Tacotron: A Fully End-to-End Text-To-Speech Synthesis Model". In: *CoRR* abs/1703.10135 (2017). arXiv: 1703 . 10135. URL: http://arxiv.org/abs/1703.10135.

[6] Madry, Aleksander et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2017. DOI: 10 . 48550 / ARXIV . 1706 . 06083. URL: https://arxiv.org/abs/1706.06083.

[7] Stoller, Daniel, Ewert, Sebastian, and Dixon, Simon. "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation". In: *CoRR* abs/1806.03185 (2018). arXiv: 1806 . 03185. URL: http : //arxiv . org/abs/1806.03185.

[8] Wilkinson, Willoiam J. et al. *Unifying Probabilistic Models for Time-Frequency Analysis - URL -* . 2018. DOI: 10.48550/ARXIV.1811.02489.

[9] Purwins, H. et al. "Deep Learning for Audio Signal Processing. - URL - ". In: *IEEE Journal of Selected Topics in Signal Processing, Selected Topics in Signal Processing, IEEE Journal of, IEEE J. Sel. Top. Signal Process* 13.2 (2019), pp. 206–219. ISSN: 1932-4553.

[10] Jordan, Michael I. "Artificial Intelligence—The Revolution Hasn't Happened Yet - URL - ". In: *Harvard Data Science Review* 1.1 (July 1, 2019). DOI: 10.1162/99608f92.f06c6e61.

[11] Rebala, Gopinath, Ravi, Ajay, and Churiwala, Sanjay. *An Introduction to Machine Learning. [electronic resource].- URL -*. Springer International Publishing, 2019. ISBN: 9783030157296.

[12] George B. Dantzig, Web.archive.org. *The Nature of Mathematical Programming URL,* 2022.

[13] Chollet, François. *Deep Learning with Python. - URL -*. Manning, 2021. ISBN: 9781617296864.

[14] Willamette.edu. *Momentum and Learning Rate Adaptation URL*. 2022.

[15] Brownlee, Jason. *Gradient Descent With RMSProp from Scratch URL*. 2022.

[16] Kingma, Diederik P. and Ba, Jimmy. *Adam: A Method for Stochastic Optimization -URL-* . 2014. DOI: 10.48550/ARXIV.1412.6980.

[17] Zhou, Zhi-Hua. "A brief introduction to weakly supervised learning -URL- ". In: *National Science Review* 5.1 (Aug. 2017), pp. 44–53. ISSN: 2095-5138.

[18] Brownlee, Jason. *What is the Difference Between Test and Validation Datasets? -URL-*. 2022.

[19] Vidhya, Analytics. *Fundamentals of Deep Learning – Activation Functions and When to Use Them?URL*. 2022.

[20] Kanal, L.N. "Perceptrons - URL -". In: *International Encyclopedia of the Social Behavioral Sciences*. Ed. by Neil J. Smelser and Paul B. Baltes. Oxford: Pergamon, 2001, pp. 11218–11221. ISBN: 978-0-08-043076-8.

[21] Basodi, Sunitha et al. "Gradient amplification: An efficient way to train deep neural networks". In: *Big Data Mining and Analytics* 3.3 (2020), pp. 196–207. DOI: 10.26599/BDMA.2020.9020004.

[22] Müller, Meinard. *Fundamentals of Music Processing. [electronic resource] : Audio, Analysis, Algorithms, Applications. - URL -*. Springer International Publishing, 2015. ISBN: 9783319219455.

[23] Brockwell, Peter J. and Davis, Richard A. *Introduction to Time Series and Forecasting. [electronic resource]. - URL -*. Springer Texts in Statistics. Springer International Publishing, 2016. ISBN: 9783319298542.

[24] Tong, Howell. *Non-linear time series : a dynamical system approach.- URL -*. Oxford statistical science series: 6. Clarendon, 1990. ISBN: 019852224X.

[25] Brownlee, Jason. *A gentle introduction to exponential smoothing for time series forecasting in Python - URL -* . Apr. 2020.

[26] Spyros, Makridakis, Evangelos, Spiliotis, and Vassilios, Assimakopoulos. "Statistical and Machine Learning forecasting methods: Concerns and ways forward. - URL -". In: *PLoS ONE* 13.3 (2018), e0194889. ISSN: 1932-6203.

[27] Crone, Sven F., Hibon, Michèle, and Nikolopoulos, Konstantinos. "Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction - URL - ". In: *International Journal of Forecasting* 27.3 (2011). Special Section 1: Forecasting with Artificial Neural Networks and Computational Intelligence Special Section 2: Tourism Forecasting, pp. 635–660. ISSN: 0169-2070.

[28] Han, Jinyu, Mysore, Gautham J., and Pardo, Bryan. "Audio Imputation Using the Non-negative Hidden Markov Model". In: *Latent Variable Analysis and Signal Separation*. Ed. by Fabian Theis et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 347–355. ISBN: 978-3-642-28551-6.

[29] Verma, Prateek et al. *A Deep Learning Approach for Low-Latency Packet Loss Concealment of Audio Signals in Networked Music Performance Applications*. 2020. arXiv: 2007.07132 [cs.SD].

[30] Martin, Charles P., Ellefsen, Kai Olav, and Torresen, Jim. *Deep Predictive Models in Interactive Music*. 2018. arXiv: 1801.10492 [cs.SD].

[31] Dorado, Fernando, Suárez, Jaime, and Torres, Alejandro. "Short-Term Load Forecasting Using Encoder-Decoder WaveNet: Application to the French Grid". In: *Energies* 14 (Apr. 2021), p. 2524. DOI: 10.3390/en14092524.

[32] Brownlee, J. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python -URL*. Machine Learning Mastery, 2018.

[33] Hyndman, Rob J. and Koehler, Anne B. “Another look at measures of forecast accuracy. - URL -”. In: *International Journal of Forecasting* 22.4 (2006), pp. 679–688. ISSN: 0169-2070.

[34] Zhang, Yingchen et al. “Chapter 16 - Predictive Analytics for Comprehensive Energy Systems State Estimation”. In: *Big Data Application in Power Systems*. Ed. by Reza Arghandeh and Yuxun Zhou. Elsevier - URL -, 2018, pp. 343–376. ISBN: 978-0-12-811968-6.

[35] Oord, Aäron van den et al. “WaveNet: A Generative Model for Raw Audio”. In: *CoRR* abs/1609.03499 (2016). arXiv: `1609 . 03499`. URL: `http : / / arxiv.org/abs/1609.03499`.

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

[37] Agarwal, K. et al. “Deep Learning based Time Series Forecasting. -URL ”. In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Machine Learning and Applications (ICMLA), 2020 19th IEEE International Conference on, ICMLA* (2020), pp. 859–864. ISSN: 978-1-7281-8470-8.

Appendix - Contents

A Code

The code below has been adapted from the openly available[28] documentation page on Time Series data [36].

A.1 TSF framework code

Used libraries:

```
#imports
import IPython
import IPython.display as ipd
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

import math
import librosa
import soundfile as sf
# Make NumPy printouts easier to read.
np.set_printoptions(precision=3, suppress=True)
%matplotlib inline

import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout
from keras.datasets import mnist
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

Data Windowing

```
class WindowGenerator():
  def __init__(self, input_width, label_width, shift,
               train_df=train_df, val_df=val_df, test_df=test_df, # !!Dataset automatically loade
               label_columns=None):
    # Store the raw data.
    self.train_df = train_df
    self.val_df = val_df
```

[28]find it at:www.tensorflow.org/tutorials/structured$_{data/time_series}$

```
    self.test_df = test_df

    # Work out the label column indices.
    self.label_columns = label_columns
    if label_columns is not None:
      self.label_columns_indices = {name: i for i, name in
                                    enumerate(label_columns)}
    self.column_indices = {name: i for i, name in
                           enumerate(train_df.columns)}

    # Work out the window parameters.
    self.input_width = input_width
    self.label_width = label_width
    self.shift = shift

    self.total_window_size = input_width + shift

    self.input_slice = slice(0, input_width)
    self.input_indices = np.arange(self.total_window_size)[self.input_slice]

    self.label_start = self.total_window_size - self.label_width
    self.labels_slice = slice(self.label_start, None)
    self.label_indices = np.arange(self.total_window_size)[self.labels_slice]

  def __repr__(self):
    return '\n'.join([
        f'Total window size: {self.total_window_size}',
        f'Input indices: {self.input_indices}',
        f'Label indices: {self.label_indices}',
        f'Label column name(s): {self.label_columns}'])


def split_window(self, features):
  inputs = features[:, self.input_slice, :]
  labels = features[:, self.labels_slice, :]
  if self.label_columns is not None:
    labels = tf.stack(
        [labels[:, :, self.column_indices[name]] for name in self.label_columns],
        axis=-1)

  # Slicing doesn't preserve static shape information, so set the shapes
  # manually. This way the `tf.data.Datasets` are easier to inspect.
  inputs.set_shape([None, self.input_width, None])
  labels.set_shape([None, self.label_width, None])

  return inputs, labels
```

```python

WindowGenerator.split_window = split_window
```

Data set generation:

```python
def make_dataset(self, data):
  data = np.array(data, dtype=np.float32)
  ds = tf.keras.utils.timeseries_dataset_from_array(
      data=data,
      targets=None,
      sequence_length=self.total_window_size,
      sequence_stride=1,
      shuffle=True,
      batch_size=32,)

  ds = ds.map(self.split_window)

  return ds

WindowGenerator.make_dataset = make_dataset
```

Compile and Fit Function:

```python
MAX_EPOCHS = 10

def compile_and_fit(model, window, patience=2):
  early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                    patience=patience,
                                                    mode='min')

  model.compile(loss=tf.losses.MeanSquaredError(),
                optimizer=tf.optimizers.Adam(),
                metrics=[tf.metrics.MeanAbsoluteError()])

  history = model.fit(window.train, epochs=MAX_EPOCHS,
                      validation_data=window.val,
                      callbacks=[early_stopping])
  return history
```

Persistence Forecast:

```python
class Continuation(tf.keras.Model):
  def __init__(self, label_index=None):
    super().__init__()
    self.label_index = label_index

  def call(self, inputs):
    if self.label_index is None:
      return inputs
    result = inputs[:, :, self.label_index]
    return result[:, :, tf.newaxis]

continuation = Continuation(label_index=column_indices['ch1'])

continuation.compile(loss=tf.losses.MeanSquaredError(),
                     metrics=[tf.metrics.MeanAbsoluteError()])
```

```

```

B Tabular results

Below a table summurazing the results of different models using the toy dataset.

Method	MSE Val (loss on validation)	MabsE Validation	MSE Test
Stupid (No change, continuation)	0.0025	0.0272	0.0016
first.reg.try with percpetron (with pretrained weights!)	0.0025	0.0273	0.0069
Linear_model (Single Window)	0.0025	0.0274	0.002
Linear (Old window)	0.0025	0.0275	0.0016
Dense (single window)	0.0026	0.0285	0.002
Dense (Old window)	0.0027	0.0305	0.002
MultiStep.Dense	0.00059806	0.0135	0.00053549
Convolution	FAIL!		
Repeat 1st try Reg with conv.window	0.00080569	0.0178	0.00061444
LSTM conv.window	0.0172	0.0803	0.0131
LSTM.wide.window	0.00080289	0.0133	0.00052778

C Other Material

C.1 Taxonomy of Time Series Forecasting Problems [37]

1. Inputs vs. Outputs: What are the inputs and outputs for a forecast?
2. Endogenous vs. Exogenous: What are the endogenous and exogenous variables?
3. Unstructured vs. Structured: Are the time series variables unstructured or structured?
4. Regression vs. Classification: Are you working on a regression or classification predictive modeling problem? What are some alternate ways to frame your time series forecasting problem?
5. Univariate vs. Multivariate: Are you working on a univariate or multivariate time series problem?
6. Single-step vs. Multi-step: Do you require a single-step or a multi-step forecast?
7. Static vs. Dynamic: Do you require a static or a dynamically updated model?
8. Contiguous vs. Discontiguous: Are your observations contiguous or discontiguous?

In the case of this research the taxonomy results in:

C.1.1 Input and output

Input: Amplitude of time samples
Output: Prediction of one or more time samples amplitudes

C.1.2 Endogenous variables

Recall by Endogenoous it is meant to indicate variables whose inputs are impacted by other variables in the system and whose output variables are dependent).
In the case of this research it is clear that the variables are endogenous; This

has the interesting consequence that feature extraction is applicable and may be beneficial as a sort of data augmentation(adding to waveform other features eg. Tempo tracking, pure frequencies)

C.1.3 Unstructured data

The data is considered to be unstructured as music audio tracks do not contain clear trends, or seasonal-periodic behaviours in the traditional time series sense, i.e. classical decomposition is not applicable (see section 2.4.1). It is however fair to notice that significant correlation within different songs parts and tempo-tracking patterns can be considered structures which could aid in the study of the time series.

C.1.4 Classification vs Regression

While the initial part of the research framed the task as a classification problem, most of the analysis have been conducted with as a Regression task. This is in line with the desired final outcome.

C.1.5 Univariate Forecasts

Most of the research has been conducted with the objective to monitor a single variable across time (it is suspected that if the net can generalize between different kind of music audio tracks, then it has no necessity of getting for example audio ch1 and audio ch2 from a given song, or other kinds of data augmentation)

C.1.6 Dynamic

Ideally the final product would continue to improve after having performed analysis on audio samples. Hence, the ideal case scenario requires a dynamic model. Yet within this research algorithms have been developed as static.

C.1.7 Contiguous

It is possible to consider the constant sampling rate of a music audio track as a proof that observations are made uniform over time

C.1.8 Shallow vs Advance ML architectures

This is another thematic confronted during the scope of the research. It would prove useful to carry on the study here conducted on more sophisticated ML architectures in order to understand whether shallow methods actually ultimately outperform more advanced ones.

CHALMERS
UNIVERSITY OF TECHNOLOGY

musictribe