POLITECNICO DI TORINO

Master's Degree in Communication and Computer Network Engineering

Master's Degree Thesis

Packet Loss Concealment in Networked Music Performance using Auto-Regressive Models

Supervisors

Prof. Andrea BIANCO

Prof. Cristina ROTTONDI

Candidate

Yuen HUANG

March 2022

Abstract

A Networked Music Performance (NMP) is a real-time Internet application, in which different performers or musicians hold the music show remotely and interact with each other going through communication networks. For enabling musicians to play simultaneously from distance, a low latency transmission is strictly required. Therefore, to minimize the end-to-end delay, uncompressed, bidirectional audio streams are used in some typical implementations of NMP applications, along with User Datagram Protocol (UDP) running at transport layer. However, due to the best effort mechanism of the Internet, it is hard to avoid the network jitter and data loss. And UDP does not support acknowledgement and re-transmission mechannism to recover the lost or excessively delayed data. Thus, exploring packet loss concealment (PLC) approaches in NMP applications to solve this problem, in order to make NMP realistic with a high quality for musicians and audience, is becoming a challenging topic in the last decades for musicians and engineers.

A feasible way to implement PLC is to synthesize the lost music data, in real time, that's used to inserted back into the audio streams to be played out. The algorithms for prediction of future music data, that could be possibly lost, based on the historic data are worth to be investigated, because the correctness of the music synthesis is of pivotal importance.

In this thesis, we propose a PLC method relying on the state-of-the-art Autoregressive (AR) Models, which were developed by statisticians firstly to do time series forecasting. We use this technique to predict the future music data that could be lost and analyse its performance by adjusting the related parameters for different genres of music.

Acknowledgements

I must take this opportunity to thank all the support I received during my Master journey, without which this thesis could not have been possible.

First of all, I would like to express my greatest appreciation to my supervisors, Prof. Andrea Bianco and Prof. Cristina Rottondi, who have been being patient and generous to me throughout this project. I am grateful for they gave me a chance to pick a thesis topic that is suitable for me to improve my personal skills.

Then, a big thank goes to Dr. Matteo Saccheto for assisting me throughout the project and giving me valuable advice and critical suggestions for my work.

What's more, I would like to express my gratitude to all my friends who have helped me during these years. A very special thank to my classmates Ian and Li Cheng, who worked with me and helped me a lot for many difficult projects and exams.

Finally, I want to express my deepest love to all my families and my boyfriend, Jie ZHAN. Though we were physically apart for most of the time in the past years, you have always been connecting with me and supporting me throughout the hardest periods.

Life is still going on, and I am on my way to a new adventure. Best wishes to all the people and take care!

Table of Contents

List of Tables

List of Figures

Acronyms

NMP Networked Music Performance

AR Autoregressive

PLC Packet Loss Concealment

TCP Transmission Control Protocol

UDP User Datagram Protocol

IP Internet Protocol

ADF Augmented Dickey-Fuller

ACF Auto-Correlation Function

PACF Partial Auto-Correlation Function

MAE Mean Absolute Error

RMSE Root Mean Squared Error

Chapter 1

Introduction

Networked Music Performance (NMP) is an interactive application with a tremendous potential impact on professional and amateur musicians, as it enables real-time interactions from remote locations. At the time of this writing, the Covid-19 pandemic is widely spreading all around the world and social distancing is required, as well as mobility restrictions between cities or countries are carried out. Thus, the implementations of NMP have been gathering increasing attention. The goal of developing NMP applications is to revolutionize the traditional concept of live musical performance. To ensure a good quality of experience in an implementation of NMP for both musicians and audience, very strict requirements must be satisfied to keep one-way end-to-end transmission latency below a few tens of milliseconds [1]. Using uncompressed audio streams and sending UDP (User Datagram Protocol) audio packets directly between different cites is a common solution to reduce time overhead in a real time application. Thus, the processing time of compression codec and the delays introduced by the re-transmission mechanism implemented by TCP (Transmission Control Protocol) can be avoided by sacrificing data transfer reliability.

However, the most critical limiting factor of distributed networked performances is the impact of the data loss and delay jitter introduced by IP networks, which is unavoidable. A lost packet causes a gap in the play-out audio and an excessively delayed packet is treated as a lost packet. Therefore, alternative recovery mechanisms must be devised to deal with packet losses in order to make the audio glitches not perceptible. Packet loss concealment approaches mainly aim at finding a way to fill the gaps introduced by losses in the play-out audio to ensure the quality of experience in the distributed music performance.

In this thesis, we propose a Packet Loss Concealment (PLC) method which is implemented by using Autoregressive (AR) Models. Our goal is to explore the AR models to do music forecasting, for different genres of music, as well as identify a trade-off with time consumption, to find appropriate combinations of the parameters to get a good quality of performance.

AR model is a special case of multi-linear regressive model, in which the forecast variable can be obtained by using a linear combination of its past values, i.e. an AR model predicts a variable of interest against the lagged values of itself. Our idea is to predict and synthesize the lost data sections in the audio streams against their historic data by leveraging AR models, and then to insert them back to fill the gaps in music play-out, to increase the quality of the experience in NMP.

AR models can only be applied to stationary time series, thus we use tools relying on the Augmented Dickey-Fuller (ADF) test as our stationarity proof for music samples before doing the experiments. Our data set is a collection of 28 music files, consisting of 22 music files of five genres and six mixture music songs, on which we will do the iterations for experiments. All of the audio files are sampled at the same rate, the number of samples of each file is at least 50000 and can reach 5000000.

In our project, our experiments are mainly focused on exploring and evaluating the performance of music data forecasting instead of building a regular VOIP communications system, i.e. training the model to do music forecasting and collecting quantitative measurements of the prediction performances. Thus, for each audio file, we randomly pick hundreds of positions to mock the data losses that would occur in a real scenario. AR models are trained and then used for prediction by adjusting various parameters. In this project, the training size and lag number are the two most relevant parameters. The training size defines how much historic data of the music series we use, and the lag number indicates how many lagged values of the data itself are taken into account to train the model. By changing these sizes and combining them in different ways for different kinds of music prediction, we may have different levels of quality of performance of our models. The larger the values are, the more time consumption for the whole training and predicting procedure is required. The test size, another interesting parameter, is actually the size of the "packet" we use to transmit, thus the size of data losses, in the real-time NMP application. It may depend on many complicated factors such as hardware resources, network condition, etc, but here in this project we are only interested in the effects of changing this size on our AR models.

At last we analyse the performance in a quantitative way by adopting two reliable error metrics, the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE). These two metrics are widely adopted for analysing the accuracy of linear regression models for time series.

The main observation in this thesis is that AR models are able to closely predict the future audio samples when the waveform of the audio data is showing relatively regular patterns regardless of instruments, melodies, or vocals.

The remainder of this thesis is organized as follows: after briefly reviewing the related literature and studies in Chapter 2, some background notions on AR models are provided in Chapter 3 and then the proposed prediction framework is described in Chapter 4. A performance assessment is provided in Chapter 5. Conclusions are provided in the final Chapter, the Chapter 6.

Chapter 2

Related Work

This chapter briefly describes the history and related studies of networked music performance as well as one of its key steps, i.e. packet loss concealment, which is aimed at making the performance better without perceptible sound deterioration.

2.1 History of Networked Music Performance

Over the last two decades, the continuously improving performance of broadband internet services has been bringing out the widespread diffusion of internet applications. NMP, as an internet application, is envisioned as a potential revolution of a traditional concert by enabling musicians perform together from distance.

One of the first NMP experiments was performed by John Cage in 1951 by using inter-connected radio transistors as musical instruments with the piece *"Imaginary Landscape No.4 for Twelve Radios"*. It was an innovative experiment but it was heavily constrained by the limited technologies of that time [2].

The rise of computers was making a big progress in the media communication. Therefore, in the late 1970s, one of the earliest NMP experiments with computers was performed by a band named *"The League of Automatic Music Composor"*, widely regarded as the first musicians to do live musical performance by creating networks of interacting computers and other electronic circuits. They approached the computer network as one large, interactive musical instrument which is made up of independently programmed music machines.

Another more sophisticated experiment was held between the mid-1970s and late 1990s by Chris Brown and John Bischoff [3]. Composers experimenting with microcomputers as musical instrument automata, in San Francisco Bay Area, sending MIDI data between musicians instead of audio signals over Ethernet due to the limited bandwidth at that time.

After 1990s, the arrival of the new generation of internet makes real-time audio applications running over the internet possible. Stanford University's CCRMA gives a simplified approach to high quality music and sound over IP network. It suggests data compression to be eliminated to reduce delay and enhance signal-quality. TCP/IP (Internet Protocol) is used in unidirectional flows for its delivery guarantees. By exploring the use of audio as a network measurement tool, SoundWIRE (sound waves over the internet from real-time echoes) created a sonar-like ping signal to display to the ear qualities of bidirectional connections [4].

In 1990s, an experimental applications called Distributed Immersive Performance (DIP) is carried out. DIP aims to develop the technology for live and interactive musical performances in NMP by interconnecting different physical locations with very high fidelity multi-channel audio and video links. The system is capable to run over local area networks and internet under the utilization of novel protocols which are for the low latency and synchronized real-time delivery of media data [5].

Furthermore, there are still other groups and projects deadling with the research on NMP. The project jacktrip [6] is used for concerts and online jamming. And it includes SoundJack, an uncompressed audio streaming application, which allows low latency multimedia communications on standard computer systems instead of dedicated hardware [7]. DIAMOUSES is a research and development project whose goal is to enable many different application scenarios for NMP including master teaching, show rehearsing, jamming sessions and collaborations [8].

Moreover, in the recent years, different network architectures, such as client-server [9] [10], master-slave [11] and decentralized peer-to-peer infrastructures [12] [13], have been investigated as enabling design paradigms for NMP systems.

2.2 Analysis on Cognition and Perception of NMP

According to the article [14], the estimated delay threshold was 25ms corresponding to a maximum separation of 8.5 meters to ensure the maintenance of a common tempo without a conductor.

An extensive evaluation experiment was carried out by [15]. Based on the musical features of various music instruments and music genres, this paper investigates the tolerance of remotely interacting musicians towards adverse network conditions. The article mentioned the sound qualities of the instruments have a significant effect on the perceptual quality of the performance. In addition, the delay tolerance threshold is not only depends on the total network delay, but also on the role and the timbral characteristics of the musical instruments that are involved, as well as the rhythmic complexity of the piece being played.

In article [16], the authors provide the experimental results convincing the readers that a latency of 20-30ms is acceptable for most multimedia applications. However, in article [17], the perception of asynchrony in musical performs under different situations is studied, and in a normal musical performance, asynchronies up to 50ms in supposedly simultaneous notes are common. The latency tolerance in music performmances also strongly depend on the piece of music and instruments [5]. And due to the delay between the action of pressing the key and the onset of the note is about 100ms for quiet notes and 30ms for forte notes [18], the tolerance for piano music performances can reach more significant values.

2.3 Packet Loss Concealment Algorithms in NMP

Making use of low-latency uncompressed audio streaming technologies makes networked music performances across the Internet a practical application. However, due to the best-effort paradigm of Internet, we can't avoid to have packet losses and delay jitters which may insert glitches into the music play-out. So the adoption of PLC techniques in NMP is very much necessary.

The majority of the studies on PLC in NMP applications focus on recovery methods for the transmission of MIDI signals instead of audio data. The authors of [19] proposed a PLC method by leveraging a recovery journal section within the

RTP packet which holds the latest music status information so that a receiver can recover from a wrong status due to previous lost packets. Still for NMP performed with MIDI signals, other recovery approaches rely on auxiliary reliably-connected channels to transmit critical MIDI events [20] or support the acknowledgment mechanisms to allow for re-transmission of lost music data [21].

Alternatively, the prediction of the waveform of lost music audio data can be an effective PLC approach. In article [22], a state-of-the-art AutoRegressive (AR) model is used for PLC in speech. The idea of predicting audio data by means of an AR model for speech can be applied in NMP too. In a jacktrip connection mentioned above [23], its built-in PLC algorithm simply repeats the last good packet that was received. An alternative choice of its PLC mode is to mute the audio until the next good packet is able to be played out. In most cases, these methods can introduce significant discrepancies and likely the distortion is audible. These two primitive methods are referred as silence substitution and pattern replication in the previous literature [24]. As in prior work [24], these two PLC methods will be used as the benchmark comparisons of our experimental results.

Further more, the era of artificial intelligence has come so modelling the music data for future prediction in NMP applications by machine learning can be a very interesting research. The authors of article [1] proposed a deep learning approach for Low-Latency PLC of audio signals in NMP. Results in this article show that, for some cases, the ML algorithm outperforms the AR approach in repairing audio signal gaps. Because the current AR models for PLC in NMP has not been widely studied and explored, we choose AR model as our implementation in this thesis.

Chapter 3

Background

This chapter provides a background on time series forecasting using AR models, which will be exploited in this thesis to predict the lost packets in NMP. In addition, the music data used in this thesis are presented in a mathematical way and the feasibility of using AR models for predicting music data is explained.

3.1 Time Series Forecasting with Autoregressive Model

3.1.1 Linear Regression Model

The basic concept of time series regression models is that the models forecast the time series of an interest variable y_t assuming that it has a linear relationship with other time series x_t. The y_t variable is usually called a forecast variable, regressand, dependent, or explained variable. The x_t variable is usually called a predictor variable, regressor, independent, or explanatory variable. Simple linear regression is the common forecasting model in the simplest cases. It estimates a linear relationship between the regressand y_t and the regressor x_t:

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t. \tag{3.1}$$

The coefficients β_0 and β_1 denote the intercept and the slope of the line respectively. As shown with an example in (3.1), a positive slope β_1 indicates that the correlation between predicted y_t and x_t is positive, and vice versa.

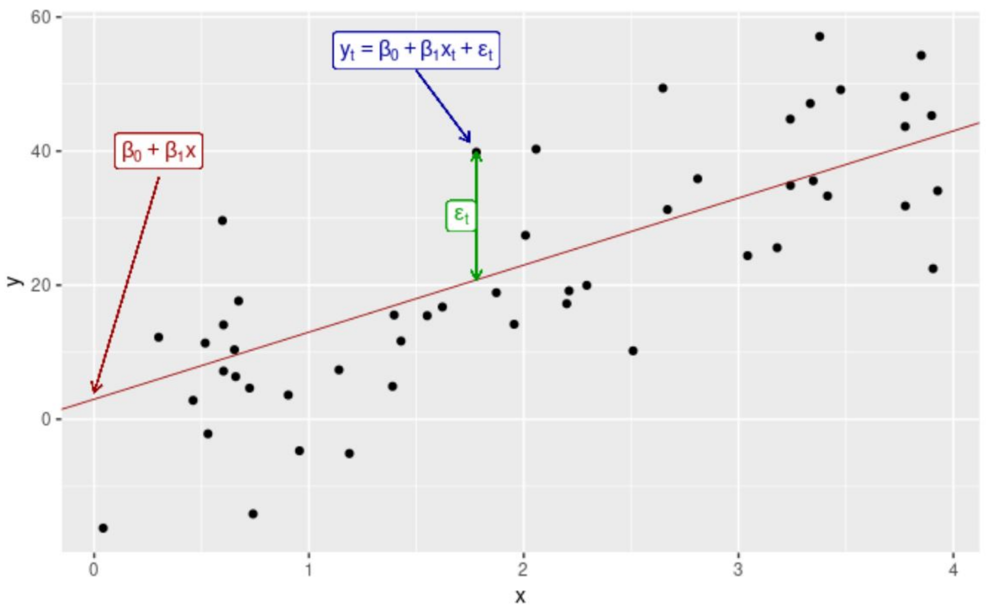

Figure 3.1: An example of data from a simple linear regression model [25].

3.1.2 Multiple Linear Regression Model

A multiple linear regression model is a linear regression model when it has two or more predictor variables. The general form of a multiple regression model with k predictor variables is expressed as:

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t. \tag{3.2}$$

The coefficients β_0,..., β_k show, in a quantitative way, the effect of each predictor $x_{k,t}$ on the forecast variable y_t after taking into account the effects of all the others. In the linear regression model, we assume each predictor x is not a random variable, which means we could control the values. And the respective coefficients can be estimated when the model is trained along with an estimated minimum error.

3.1.3 Autoregressive Model

AR model is a special case of multiple regression model. In an AR model, a forecast variable y_t can be got by using a linear combination of its past values. In other words, AR models predict a variable of interest against the lagged values of

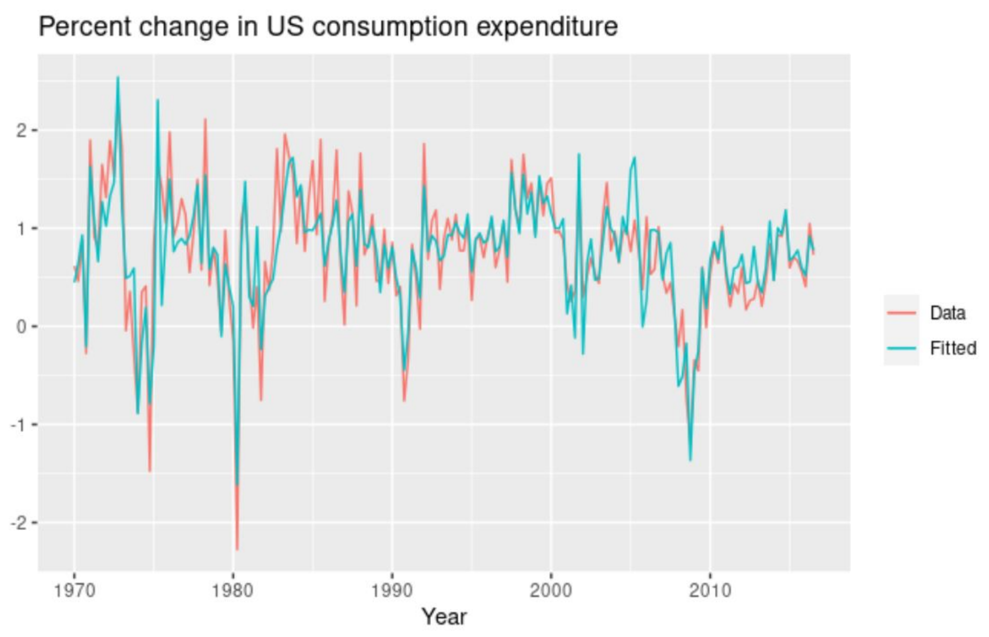

Figure 3.2: Example: Time plot of actual US consumption expenditure and predicted US consumption expenditure. [26].

itself. An autoregressive model of order ρ is denoted as $AR(\rho)$ and can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_\rho x_{t-\rho} + \varepsilon_t. \tag{3.3}$$

where ε is white noise and c is a constant. The order ρ of an AR model should be chosen by analysing the PACF of the data before building and training the model.

Stationarity and Differencing

AR model can only be applied to stationary time series forecasting. So stationarity should be examined before building a forecasting model for a time series. A time series whose statistical properties do not depend on the time is considered as stationary.

The following properties should be found throughout the entire series:

- Flat looking without trend
- No seasonality

- Constant mean over time
- Constant variance over time
- Constant correlation structure over time

If a time series is non-stationary, differencing can be used to eliminate or reduce trend and seasonality. Computing the differences between consecutive observations is known as differencing.

ACF and PACF

Exploring ACFs and PACFs can help researchers understand the temporal dynamics of a time series. An ACF measures the average correlation between data points in a time series and previous values of the series measured for different lag lengths. However, PACFs, Partial ACF, does a the similar work as ACF with each correlation under controlling for any correlation between observations of a shorter lag length.

As mentioned before, two key factors about the temporal dynamics should be examined before building and training the equation of an AR model:

- Stationarity of the data
- The order of AR model

Exploring and plotting the ACF of a time series is also an useful approach for analysing its stationarity. The ACF of a stationary time series decays towards zero relatively quickly, while the one of non-stationary time series does so very slowly. As shown with an example in Figure 3.3.

For a stationary time series that can be fitted by an AR model, the PACF plots will show some spikes at the lag lengths at which the data points correlate high. As a consequence, exploring PACF is a possible way to define the order of the AR model to be trained, as shown with an example in Figure 3.4.

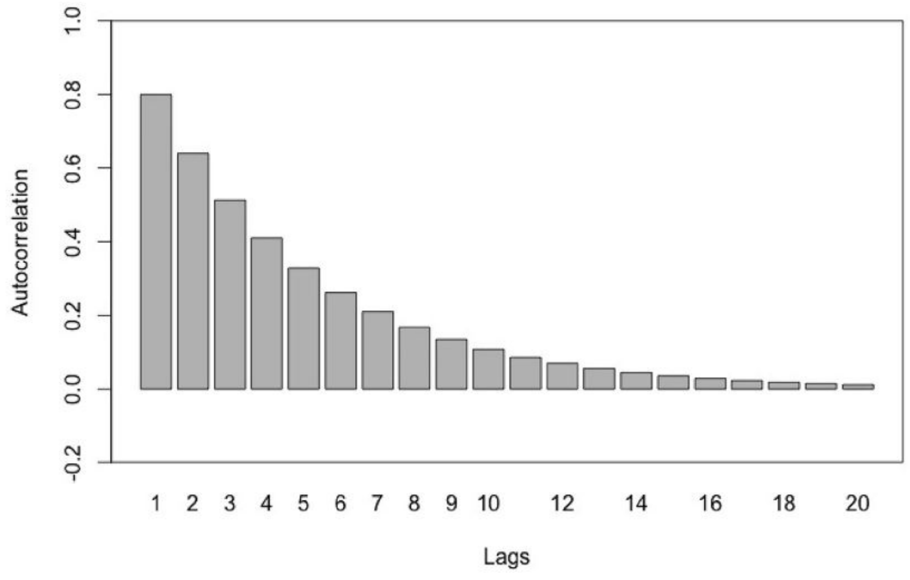

Figure 3.3: Illustrative example of an ACF with a correlation between observations at a single lag equal to 0.8. [27].

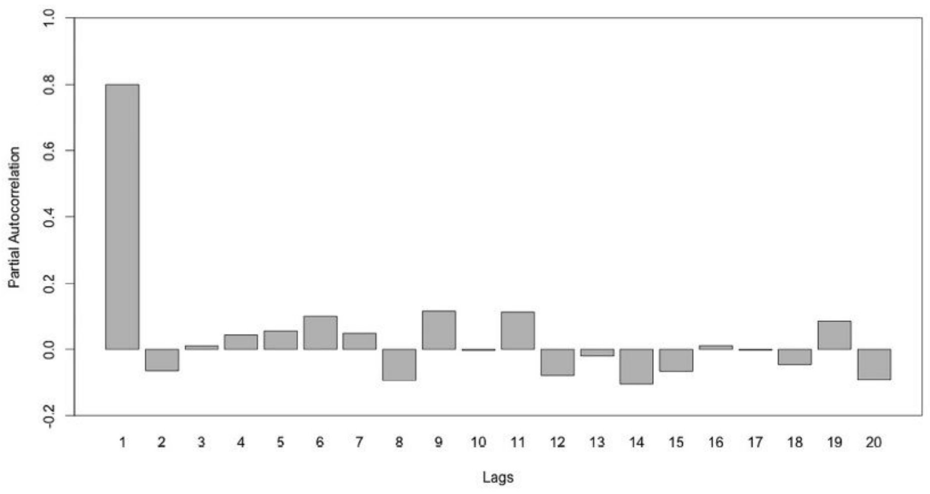

Figure 3.4: Illustrative example of a PACF with a correlation between observations at a single lag equal to 0.8. [27].

3.2 Music Data Representation

The music data set, on which the AR model will be applied to do the forecasting in this project, is all audio including digital files and real time play back. Music data, as a time series, can be represented and explained in a mathematical way, i.e. its waveform plot, to show some properties/stationarity that are under the condition of performing an AR model forecasting. An example of the waveform plot of a guitar-only music file is shown in Figure 3.5.

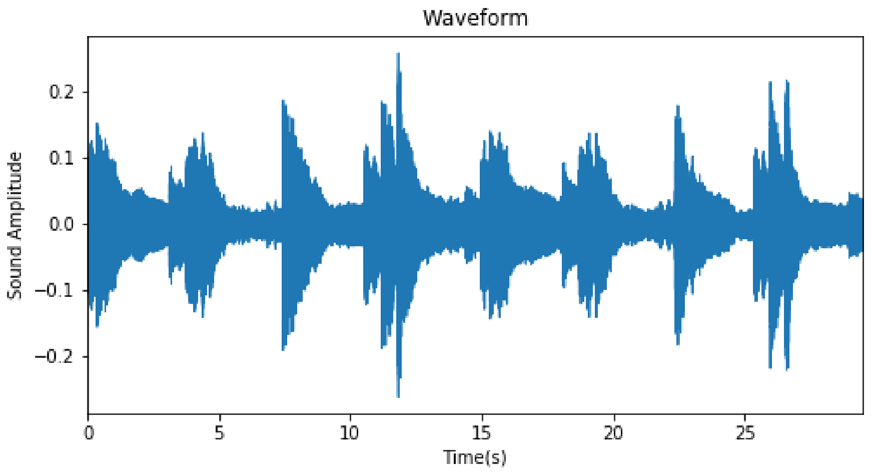

Figure 3.5: An example of waveform of a guitar-only music file.

Chapter 4

Framework

This chapter describes the framework used for processing musical data along with the implementation of auto-regressive models as our packet loss concealment algorithm in NMP.

The key concept that's worth to be recalled is that NMP is running over a computer network as a real time application, which requires strictly low latency music data streaming. As a consequence, the application should be running with UDP at transport layer other than TCP due to the fact that TCP introduces some delay to perform flow control, error control and packet re-transmission mechanism to ensure the integrity of the receiving data, while UDP without error correction mechanism is bringing a lower degree of accuracy but higher efficiency in transmission.

Therefore, to ensure a good quality of NMP in real time, error correction or PLC approaches are fairly important to be implemented in the application layer. PLC Algorithms should be smart and quickly find a way to re-create the lost data and insert back synthetic data sections into the gaps, caused by data loss or excessive delay in a best-effort communication network, in the music data streaming.

Figure 4.1 briefly depicts the overview of the system we implement. Each step will be introduced in the following sections and the most critical sub-process, our proposed solution for PLC, is *AR model training and prediction for all test positions.*

In our proposed framework, we train an AR model to predict the lost data to achieve what we just mentioned. There is no denying we need a trade-off between

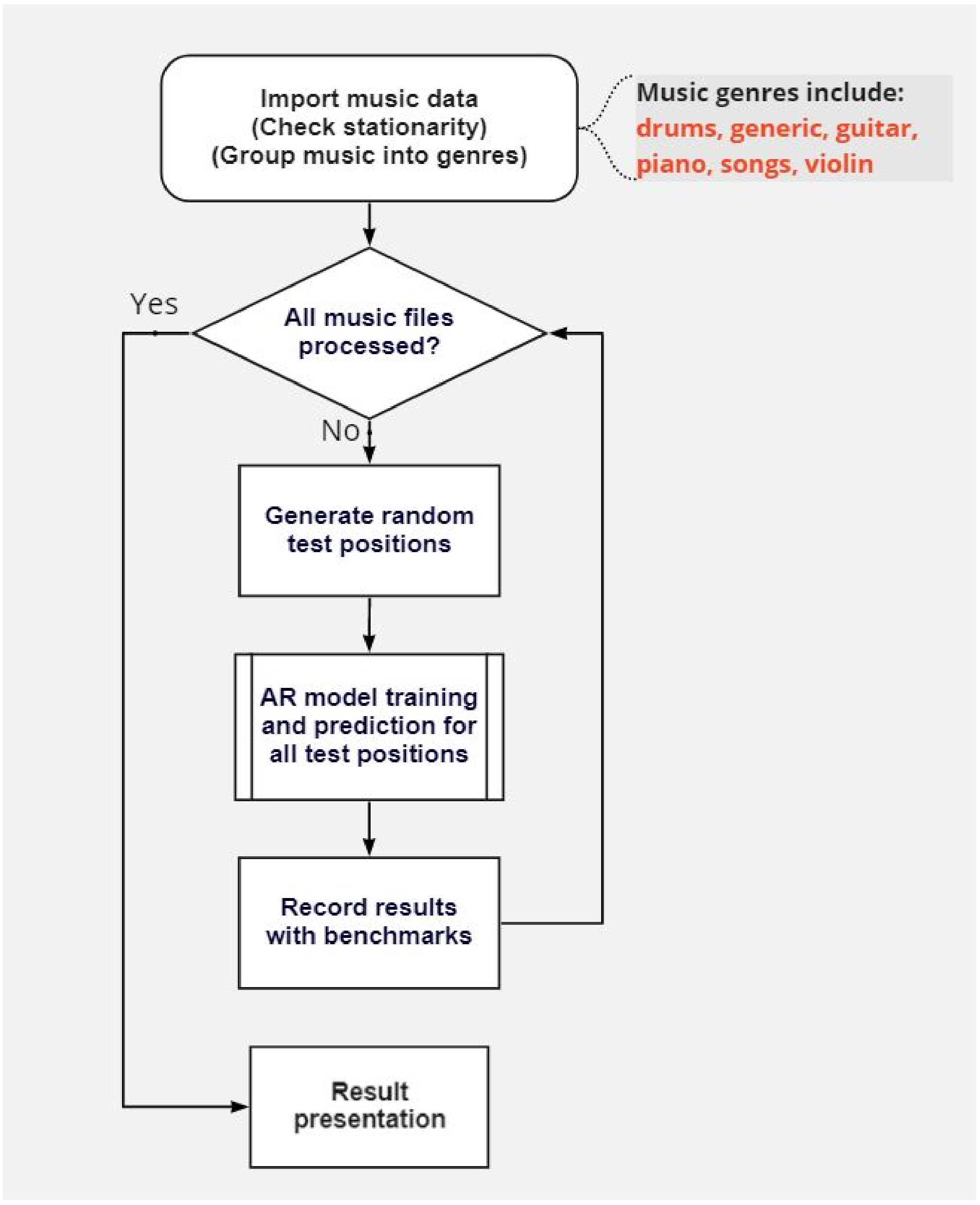
Import music data
(Check stationarity)
(Group music into genres)
Music genres include:
drums, generic, guitar, piano, songs, violin
All music files processed?
Yes
No
Generate random test positions
AR model training and prediction for all test positions
Record results with benchmarks
Result presentation

Figure 4.1: Overview of Framework

computational complexity (which means consuming more time and introducing higher latency) and prediction accuracy. So, we mainly analyse the effects on and results of our system along with AR models by adjusting different parameters.

4.1 Data Set

The data set used in this project consists of audio files in digital format *.mp3* or *.wav*, searched and collected from [28]. We separated them into five genres and a mixture: Drum, Guitar, Piano, Violin, Generic and Song. The first four kinds of music files are melodies only played by single musical instrument and the files included in Generic and Song are mainly containing vocals and a mixture of all above respectively.

4.1.1 Stationarity Proof

As we said in Chapter 3, AR models can only be applied to forecast the stationary time series. For cleaning and collecting the appropriate data set for the experiments, we need a good proof of stationarity for all music data. Here the tools we use are against Augmented Dickey-Fuller (ADF) test.

What is ADF test?

ADF test is fundamentally a statistical significance test which means there is a hypothesis testing involved with a null and alternate hypothesis. It computes and reports a test statistic and p-values, from which you can make an inference as to whether a given series is stationary or not [29].

How does ADF test work?

The ADF test is one of a category of tests called ‘Unit Root Test’, which is the proper method for testing whether the time series is stationary. Unit root is a characteristic of a time series that makes it non-stationary.

Dickey-Fuller test is a unit root test that tests the null hypothesis (α=1) in the following model equation where α is the coefficient of the first lag on Y.

$$Y_t = c + \beta t + \alpha y_{t-1} + \phi \Delta Y_{t-1} + e_t. \tag{4.1}$$

where, y_t is the value of the time series at time ’t’ and Y_{t-1} is the value at lag 1 and X_e, δY_{t-1} is the first difference of the series at time ’t-1’. The coefficient α

with value 1 is implying the presence of a unit root. If it is not rejected, the series is considered to be non-stationary.

The ADF test evolved based on what we talked about just above and acts as an augmented version of the Dickey Fuller test because the equation expands into high order regressive process in the model.

$$Y_t = c + \beta t + \alpha Y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} + \cdots + \phi_p \Delta Y_{t-p} + e_t. \tag{4.2}$$

The p-value obtained by ADF test on the music samples should be less than the significance level (usually 0.05 is taken) in order to reject the null hypothesis, which means, to prove the time series is stationary.

4.1.2 Genres and properties of Data

Drum

An example of waveform of drum music file is shown in Figure 4.2. We can see some properties of stationary here such as constant mean and variance, but the rhythm is quick which makes the prediction more difficult.

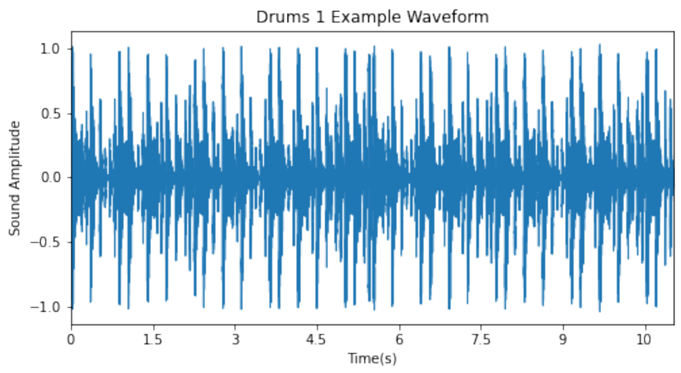

Figure 4.2: An example of waveform of the drum music

Piano

As shown in Figure 4.3, the piano music pattern looks slower and more gentle, and it is stationary. Therefore, it is likely that our AR model may predict well on piano samples.

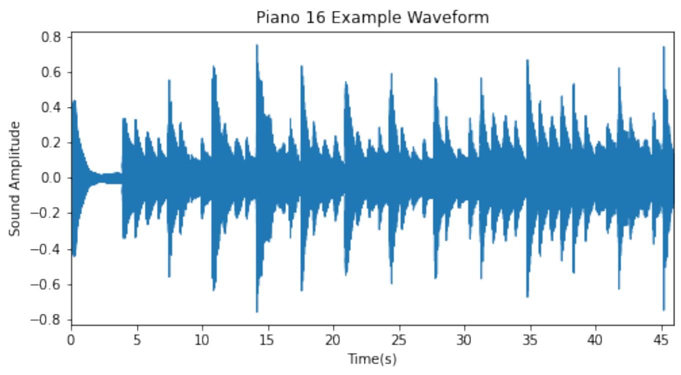

Figure 4.3: An example of waveform of the piano music

Violin

As what we see in Figure 4.3, the waveform of violin is showing a pattern which is relatively stationary.

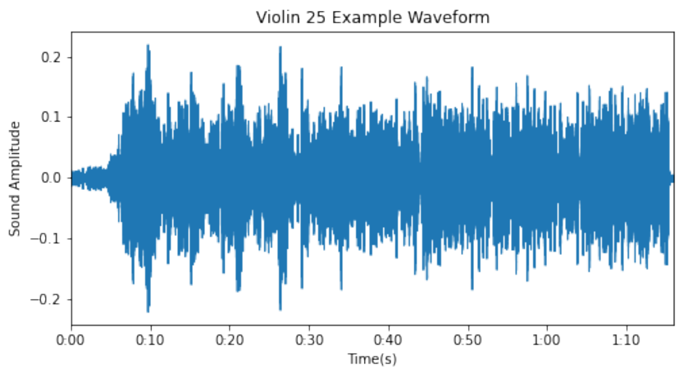

Figure 4.4: An example of waveform of the violin music

Guitar

The variance of guitar music is not a constant value, which is indicating it is not a stationary time series so our AR model would not perform very well without differencing.

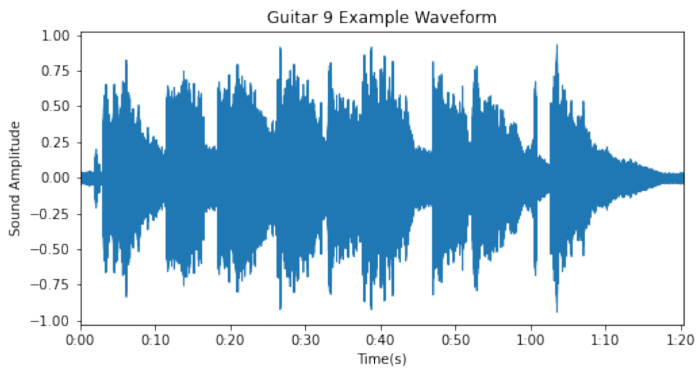

Figure 4.5: An example of waveform of the guitar music

Generic

It is not hard to find that the variance is slightly changing along time and its dispersion may infer that the AR model would have a poor performance on the generic music.

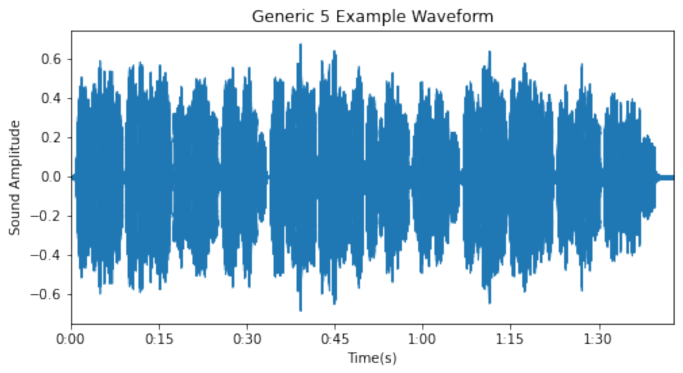

Figure 4.6: An example of waveform of the generic music

Song

We will use our best AR model, with best parameters after analysing, to train and predict a generic song which consists of various kinds of sounds played by different instruments as well as human vocals.

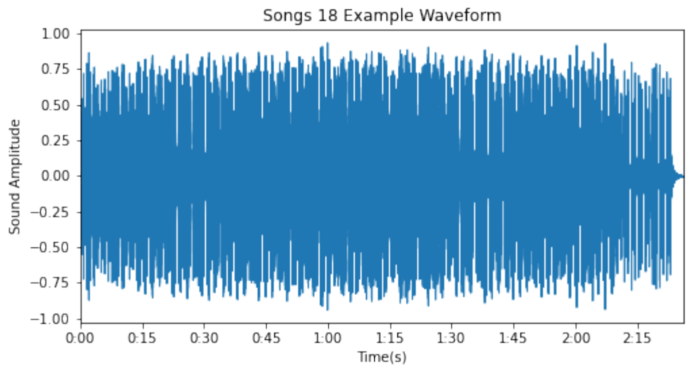

Figure 4.7: An example of waveform of a generic song

4.2 Training and Testing of AR Models for PLC

Figure 4.8 shows the framework of training and prediction of our AR model and the related interesting parameters that we need to analyse and then adjust for the best performance results.

Our experiments are mainly focused on exploring and evaluating the performance of a packet loss concealment method based on AR models, i.e. training the model and collecting quantitative measurement of the testing results, instead of building a regular VOIP communications system. So we artificially inserted gaps i.e. loss in random positions in the music time series by using some pseudo-random tools of python. Those random positions are the time indices of the end of music samples as training set and the beginning of testing samples.

The experimentation should consider different scenarios. The order of the AR model i.e. the number of lag values, is an interesting parameter that we should study, as shown in the equation 3.2, to train the model to actually get the best coefficients of each lagged value w.r.t the currently being predicted values. The higher the order of an AR model is, the more coefficients are needed to be calculated with minimum errors when training the model, which means the bigger the lag value is, the more time overhead the training procedure introduces. So obviously, the number of lag values is quite worth analysing.

Besides, the size of the training set is another critical parameter. What we want to figure out is, for every kind of musical performance, which size is the best to be use, i.e. which is the number of samples that is used to train the AR model in order to get the best performance results. An Auto-Regressive model is actually

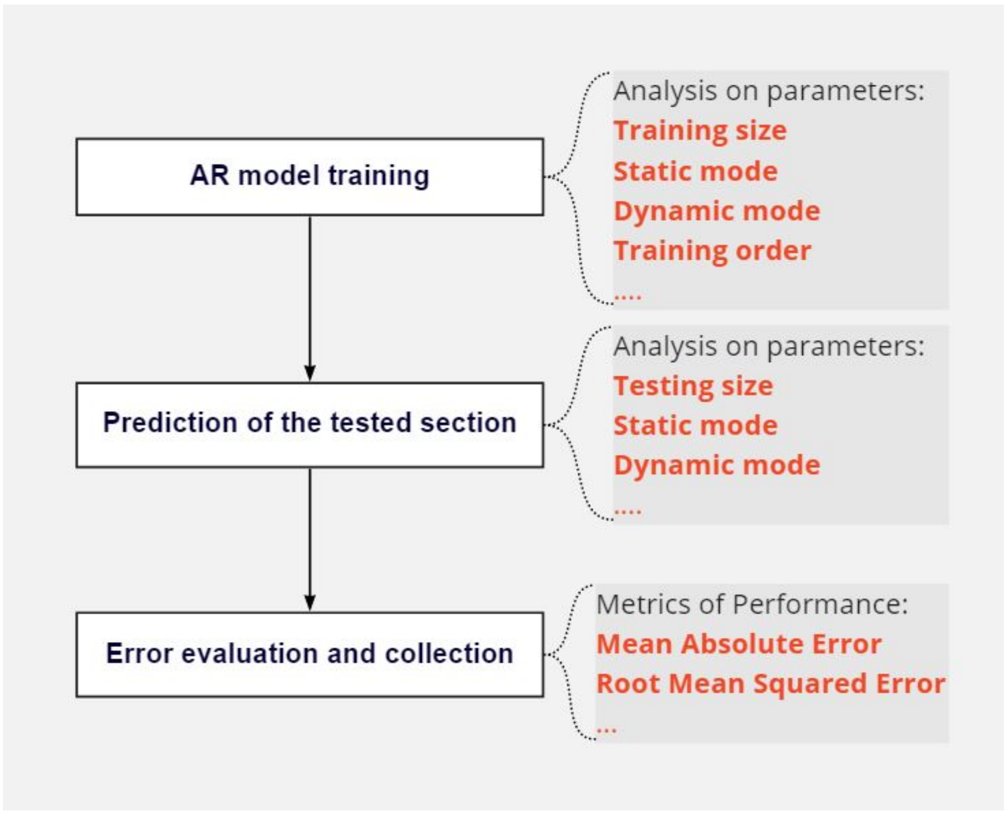

Figure 4.8: Procedure of Parameter Analysis for AR Models in PLC

using the historical data to do the forecasting for future data. But a pretty good question is: how long should we go back in time? How many historical data are actually meaningful for the data being predicted at the current test position?

By adjusting the size of the training set and the value of lag in different iterations, for each music file, we calculate the mean absolute error (MAE) and root mean squared error (RMSE) of the predicted music samples with respect to the original ones.

Furthermore, we use the latest historical samples w.r.t the test positions as the training set to train the model, and then to predict the lost audio samples.

The parameters of AR models within each iteration are fixed, i.e. for each audio file, the size of training set, the order of the model (lag values) and the length of the gap (test section) to be predicted are all unchanged for every random testing position.

Once we get a random position for testing PLC, we predict the whole patch to fill the gap. As Figure 4.9 shows, we have a stream of audio samples and randomly picked positions. In real time applications, the transmitter(receiver) is sending(receiving) media streams section by section and the section size depends on the specific application and buffer size or many other factors. Therefore, the pink section is where we do the patch prediction and performance evaluation. The training set is accumulated by the received sections in history earlier w.r.t. the test position.

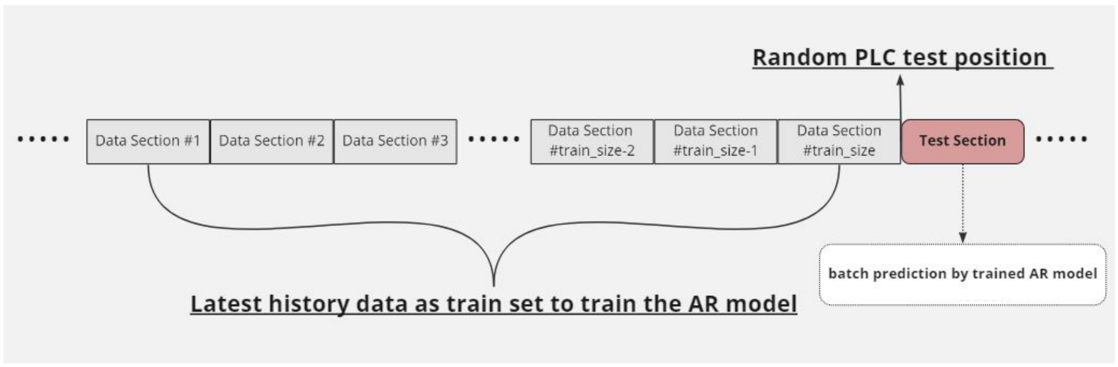
Random PLC test position
Data Section #1
Data Section #2
Data Section #3
Data Section #train_size-2
Data Section #train_size-1
Data Section #train_size
Test Section
batch prediction by trained AR model
Latest history data as train set to train the AR model

Figure 4.9: Process of Predicting Music Data by Training AR Models

4.3 Results Evaluation

4.3.1 Benchmark Methods

As we mentioned in Chapter 2, we take two approaches, silence substitution and pattern replication, as our result testing benchmarks.

Benchmark - Silence Substitution

This approach is just inserting a silent section into the gap to make a packet loss concealment.

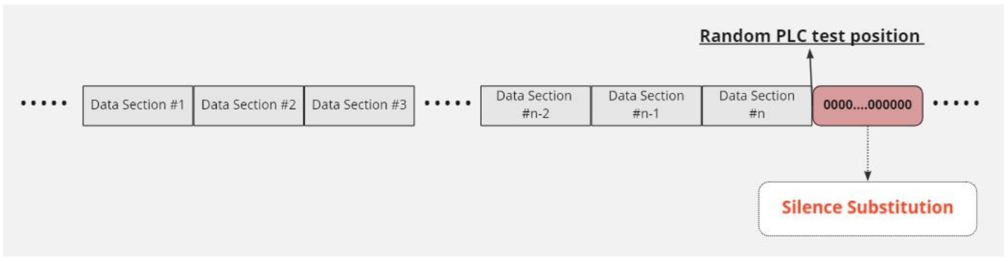

Figure 4.10: Silence Substitution For PLC

Benchmark - Pattern Replication

This approach is just repeating the last properly received data section in the lost section as its PLC mechanism.

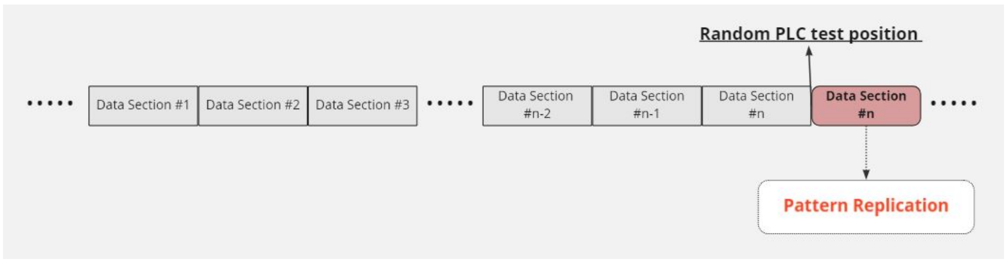

Figure 4.11: Pattern Replication For PLC

4.3.2 Performance Metrics

As we said, the experiments of this project mainly focus on analysing the performance of music data prediction by building and training an AR model as our PLC approach. So the performance metrics are playing a quite important role in the whole procedure. How good is the prediction matching up against actual values and how can we evaluate the quality of our AR model in a quantitative way? We call the difference between the actual value and the predicted one residual. For every sample, we calculated the residual and each of them will be of use in assessment. These residuals will play a significant role in judging the usefulness of a model. If the collected residuals are small, it implies the AR model does a good job at predicting that music samples, otherwise relatively poor. For statistics purposes, it is better to condense them into a single value that represents the predictive ability of our model. There are many of these summary statistics, each with its own advantages and pitfalls. In this project we decided to use MAE and RMSE as our error metrics and we'll discuss what each statistic represents.

MAE - Mean Absolute Error

The mean absolute error (MAE) is the simplest error metric for regression performance. We calculate the residual for every data point and take only the absolute value of each and then take the average of all these residuals. Effectively, MAE describes the typical magnitude of the residuals.

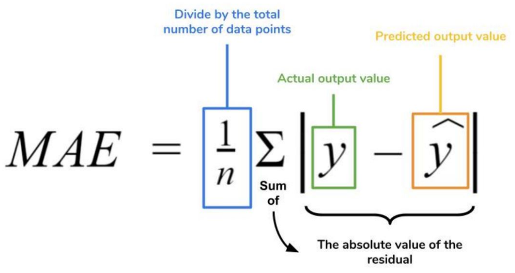
Divide by the total number of data points
Predicted output value
Actual output value
$MAE = \frac{1}{n}\sum\left|y - \hat{y}\right|$
Sum of
The absolute value of the residual

Figure 4.12: The formal equation of MAE [30]

Figure 4.13 is the graphical description of the MAE. The red lines are indicating the magnitude of the residuals. MAE is the most intuitive metric since we are just looking at the absolute difference between the actual data and predicted data. Therefore, the larger errors do not make more penalty on the result because each residual contributes proportionally to the total amount of error, which means MAE does not indicate under-fitting or over-fitting model's performance.

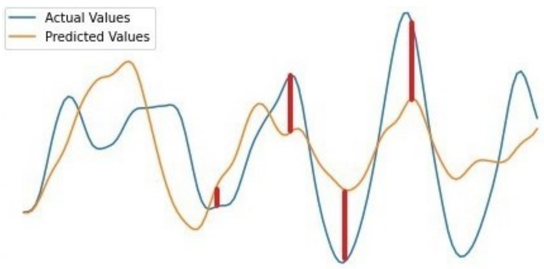

Figure 4.13: An example of comparison of predicted music data against the actual ones

RMSE - Root Mean Squared Error

As we mentioned, absolute error has its disadvantage, so squaring the residuals is more desirable when we want the outliers, or extreme values, make more penalty to the overall error. Therefore, Mean Squared Error (MSE) is squaring the difference before summing them up to get the mean value.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{4.3}$$

The big prediction errors may create glitches in the predicted music samples so we seriously want to take the outliers more into account. While each residual in MAE contributes proportionally to the total error, the error grows quadratically in MSE. This ultimately means that outliers, the predicted values greatly differing from the corresponding actual values, will contribute to much higher total error as well as penalize more our model in the MSE than they would the MAE.

Because MSE's unit does not match that of the original data. We use RMSE to convert the error metric back into similar units, making interpretation easier.

4.3.3 Time Consumption

Time consumption is another important factor that we need to take into account. We calculate the time consumption of training and prediction separately of our AR models, as Figure 4.14 shows below.

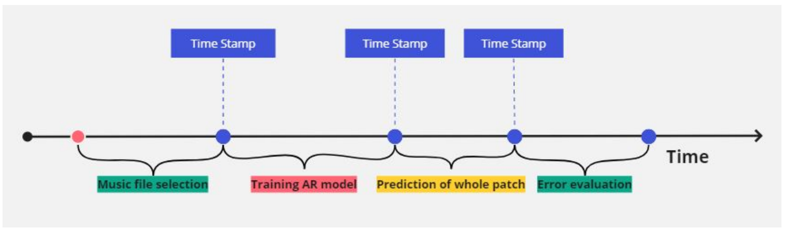

Figure 4.14: Time Consumption Calculation for Training and Prediction

Chapter 5

Experimental Results

This chapter will describe in details how the experiments are set up, as well as the results of training and predicting. Besides, some discussion will be made at last.

5.1 Experimental Setup

As explained in Chapter 4, we analyse various parameters and different modes of our AR model in training and predicting. First of all, for picking up randomly the PLC test positions, we set the python.numpy pseudo-random seed as 1 before all iterations start in order to make sure every time we run them we will get the same sequence of random integers. Then we have totally 28 music files, consisting of 22 music files of five genres and six mixture music songs, on which we will do the iterations for experiments. All of the files are sampled at the rate 22050 Hz, and the total numbers of samples of each of them are in the range from 57200 to 5159700, shown in the column named "Length" in Table 5.1.

In Table 5.1, the *music ID* is the identity number of the specific file in our experiment and it has a *Freesound ID* if it's downloaded from [28]. *Length* indicates the total number of samples that audio file has and *Format* shows the digital formats in which the music samples are saved. The songs with music ID from 0 to 5 do not have *Freesound ID* because they are downloaded from the Internet instead of *Freesound*. Their names are shown in Table 5.2. Each song is a mixture played by various kinds of instruments, so we are able to use them to evaluate the AR model performance on complex music.

Music ID	Genre	Freesound ID	Length	Format
0	Songs	-	5139072	mp3
1	Songs	-	1878912	mp3
2	Songs	-	2011968	mp3
3	Songs	-	3485376	mp3
4	Songs	-	1801152	mp3
5	Songs	-	3227791	mp3
6	Piano	171326	1698862	wav
7	Piano	570347	1058400	wav
8	Piano	164718	954703	wav
9	Piano	394027	1089375	wav
10	Piano	570169	1014300	wav
11	Violin	276887	2953216	wav
12	Violin	377258	810338	wav
13	Violin	351269	1676856	wav
14	Violin	395892	4483853	wav
15	Drum	341980	243487	wav
16	Drum	353081	5159700	mp3
17	Drum	385676	235353	wav
18	Drum	149952	403201	wav
19	Guitar	389401	278045	wav
20	Guitar	37200	1775025	mp3
21	Guitar	591737	211680	wav
22	Guitar	148695	2456186	wav
23	Guitar	614341	248063	wav
24	Generic	361685	343171	wav
25	Generic	262274	2268476	wav
26	Generic	136777	57200	wav
27	Generic	529844	128759	wav

Table 5.1: Music Data Set

For the size of training, we set it to 512, 1024, 2048, 4096 and 8192 samples each time. And for the order of the AR model, the number of lagged values that may correlate high with the value being predicted, is exponentially increasing from 1 to 128 with the power to two. What's more, the number of samples that AR models predicts, i.e. the AR model performance test size, is 64 or 128, which means we

Music ID	Song Name	Source
0	morning-garden-acoustic-chill-15013.mp3	[31]
1	jazzy-abstract-beat-11254.mp3	[32]
2	sexy-fashion-beats-11176.mp3	[33]
3	madirfan-unreleased-demo-15-01-2022-14254.mp3	[34]
4	commercial-rock-beats-11249.mp3	[35]
5	bensound-ukulele.mp3	[36]

Table 5.2: Mixture Songs

mock losing a patch of samples of length 64 or 128 each time when we are at the random PLC test position. So our goal is to explore the best combination of these parameters with specific values that we can use to train the AR models as good as possible to do music forecasting with a high quality of performance.

5.2 Results and Error Evaluation

As shown in Figure 5.1 and Figure 5.4, when the lag values increase, erratically high MAE and RMSE can appear sometimes. For example, with 128 lags and a 512 training size, the MAE and RMSE can go up to around 6.5 and 15 respectively. In such cases, a weird observation is that the endpoints of the confidence intervals can be negative. The reason for this is that the confidence intervals of the mean are calculated with an assumption of Student's T Distribution, which could produce negative endpoints with large data variance. The huge ranges of confidence intervals may imply that our AR models could have uncertain performance on musics that have volatile waveform patterns, e.g. Drums.

For the music files Drum 15 and Drum 17, unexpectedly we have worse performance when the lag value increases and the train size decreases. The partial zoomed-in plots of Figure 5.1 and Figure 5.4 are shown in Figure 5.2 and Figure 5.5 respectively. The errors with the small lag numbers are located around the ones of the benchmark 0, which indicates our AR models are not implementing any effective predictions.

Compared to them, the results of Drum 16 shown in Figure 5.3 are slightly better than other music files, but still, they are not so optimistic.

Similarly for Drum 18 in Figure 5.6, even though the performances seem better when lag numbers increase, but in the partial zoomed-in plot Figure 5.7 we can see

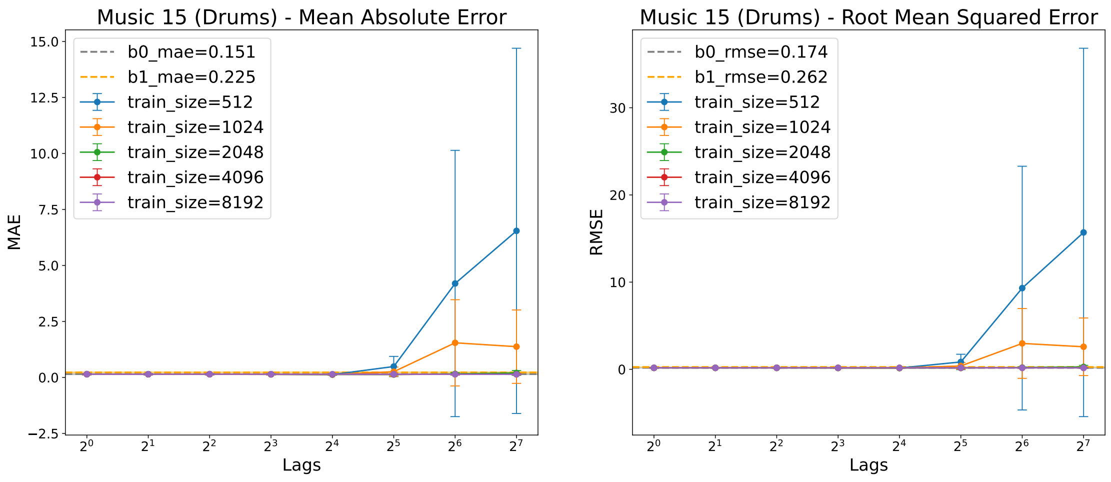

Figure 5.1: Drum - ID:15 MAE and RMSE

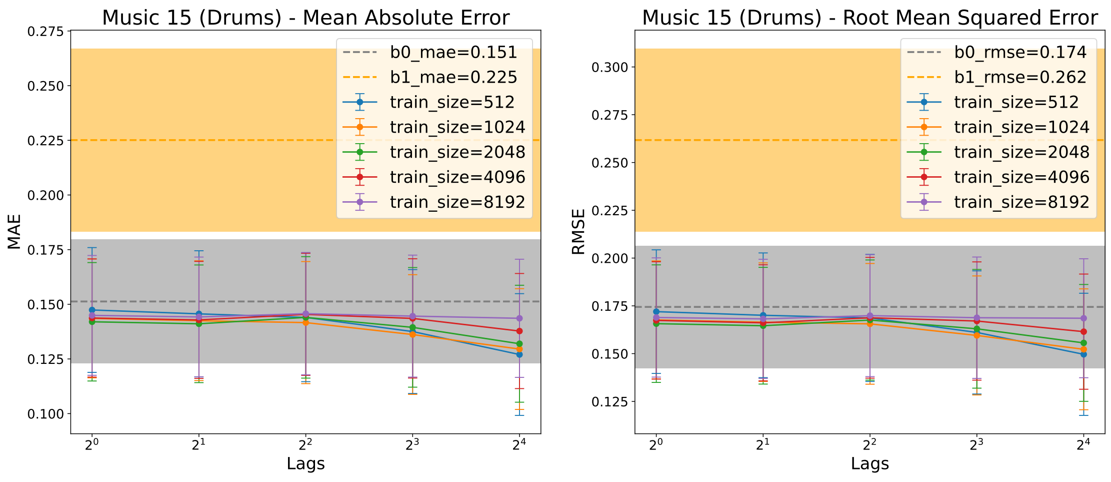

Figure 5.2: Zoomed-In Plot of Drum - ID:15 MAE and RMSE

for most of the cases, the RMSE patterns are extremely irregular and even worse than two benchmarks.

Examples of comparison are shown in Figure 5.8 and Figure 5.9, the predicted values of the music files are not close to the actual values. Our AR models have poor performances on all the drum music files.

For guitar music files from 19 to 22 shown in Figure 5.11,Figure 5.12,Figure 5.13 and Figure 5.14, the RMSE are going down from around 0.2, 0.12, 0.14 and 0.1 to around 0.06 respectively when the lag numbers increase. After lag number 32, the

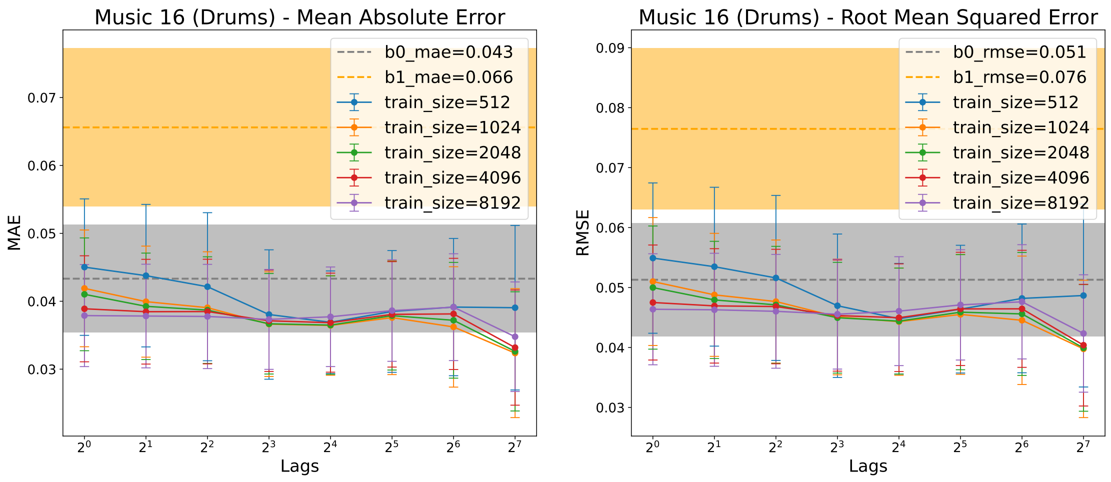

Figure 5.3: Drum - ID:16 MAE and RMSE

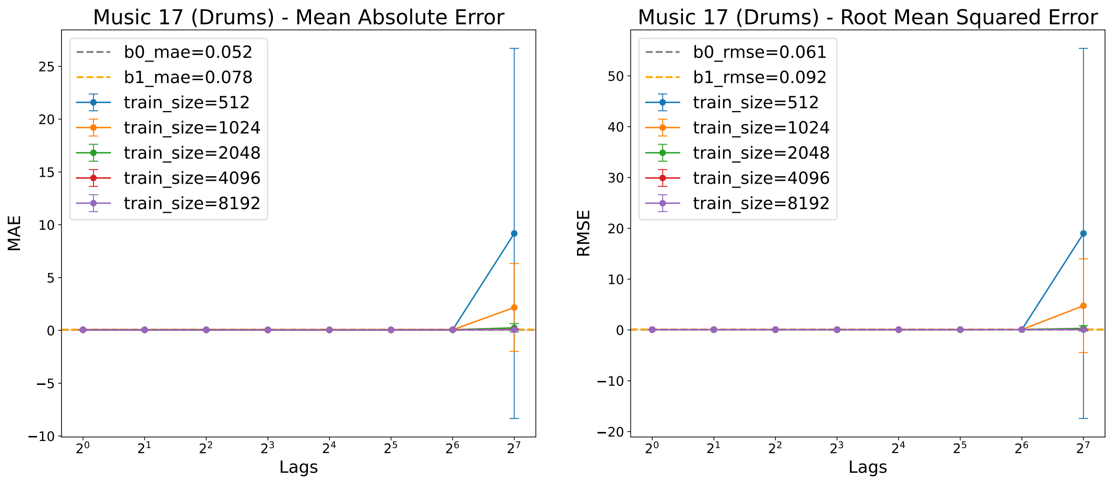

Figure 5.4: Drum - ID:17 Error versus Time Overheads

model with any training size can outperform the benchmark approaches. This is a sign that our AR models are likely able to have a good prediction performance on this kind of guitar music.

But we find some weird irregular behaviours of MAE and RMSE in Figure 5.15. After lag 8, volatile patterns are shown in both MAE and RMSE. Refer to zoomed-in plot in Figure 5.16 for more details and we can see before lag 8, the errors are all located around the ones of benchmark 0.

One example to show the reason why we get these is in Figure 5.17 comparing the

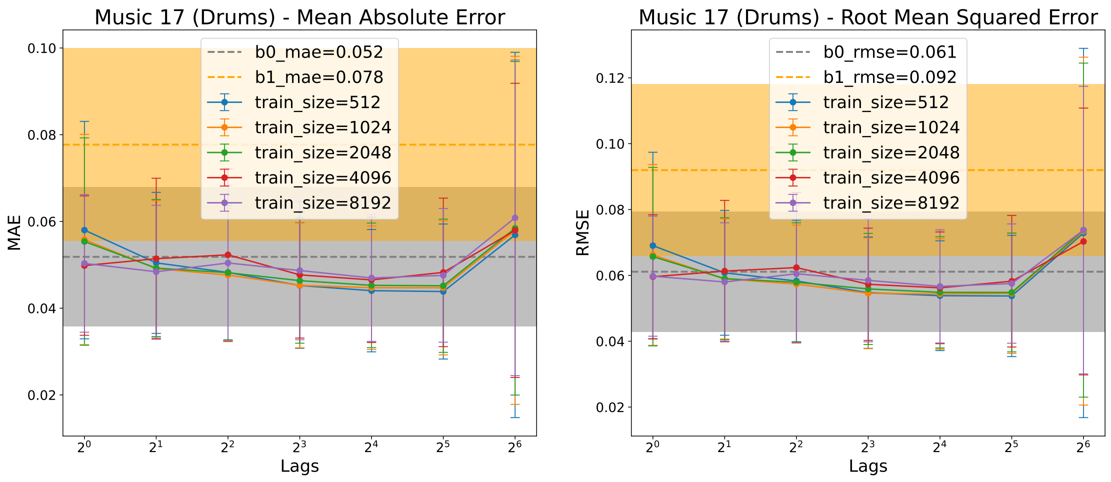

Figure 5.5: Zoomed-In Plot of Drum - ID:17 MAE and RMSE

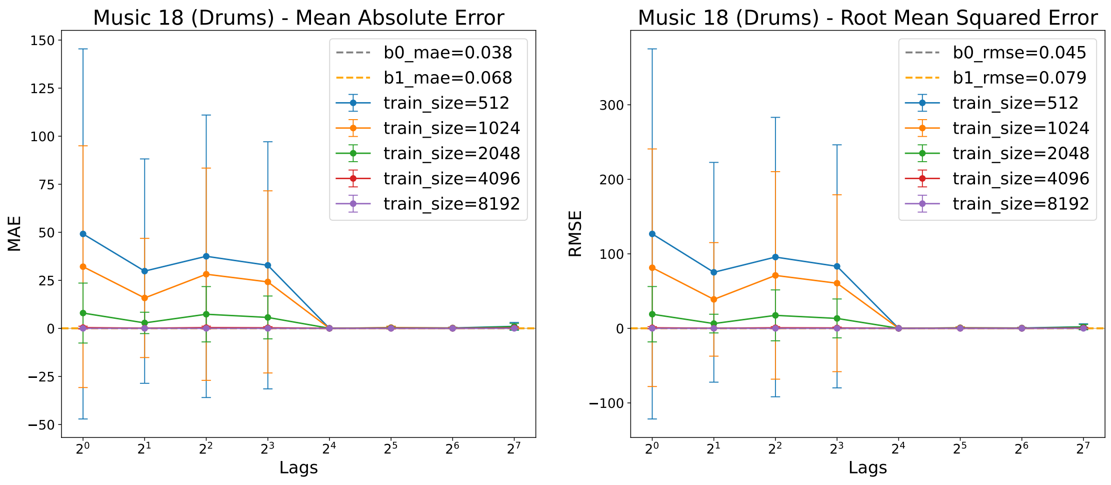

Figure 5.6: Drum - ID:18 MAE and RMSE

actual values and predicted values of Guitar 23. We may say that for the music file Guitar 23, our AR models are likely not doing any effective predictions.

Based on the results shown in Figure 5.19, Figure 5.20, Figure 5.21, Figure 5.22 and Figure 5.23, we can find that for piano music with any training size, the RMSE of our AR models can go from around 0.12 to 0.04, or from around 0.06 to 0.02, when the lag numbers increase from 1 to 128. No erratic error patterns are found in these music files because the errors are always decreasing when the models are trained at a higher order. And in many cases, we just need the lag number to reach 8 and we can have a better performance against the benchmarks.

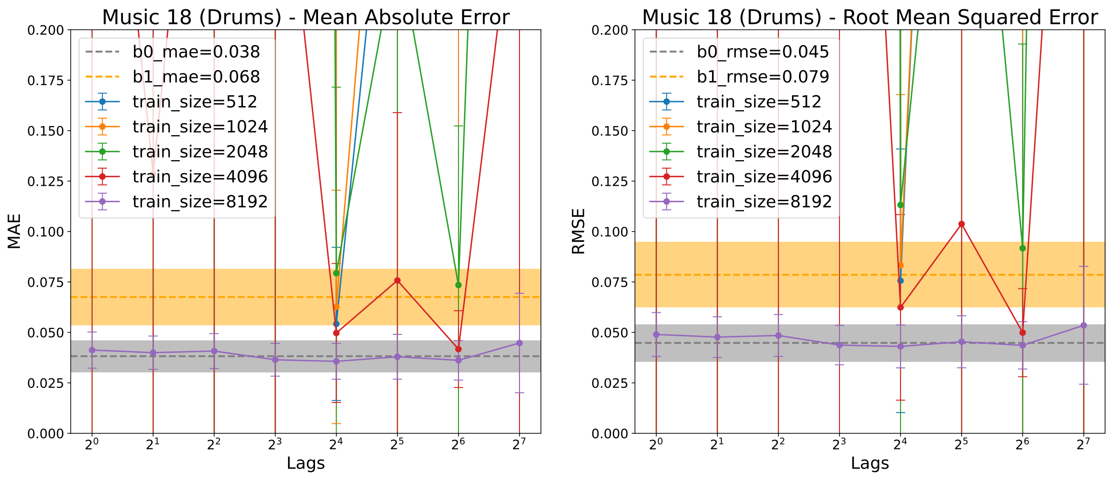

Figure 5.7: Zoomed-In Plot of Drum - ID:18 MAE and RMSE

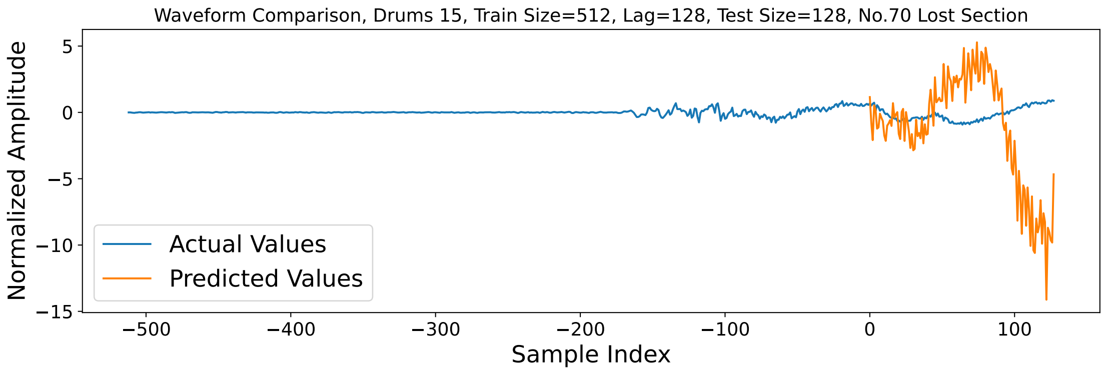

Figure 5.8: A waveform Comparison for Drum ID:15 at the No.70 lost section

The models for violin music files have similar behaviours as the ones for piano music. In many cases, the errors are going down when the lag numbers increase. Except violin 14 in Figure 5.27, the RMSE of violin 11, violin 12 and violin 13 can go from around 0.04 to around 0.01, from around 0.08 to around 0.01 and from around 0.03 to 0.15 respectively. Violin 11 and 12 outperform benchmarks more than the ones of violin 13 and 14. In Figure 5.27, except the models with training size of 512 and 1024 have irregular trends after lag 32, other cases still have the slightly better performance against benchmarks when lag numbers increase.

We have seen the results for music files in genre of drum, guitar, piano and violin and all of them are single-instrument melodies or sounds. In the following we are going to check the results for generic music files, mainly containing vocals, and the generic songs, mainly containing a complex mixture of instrument sounds and

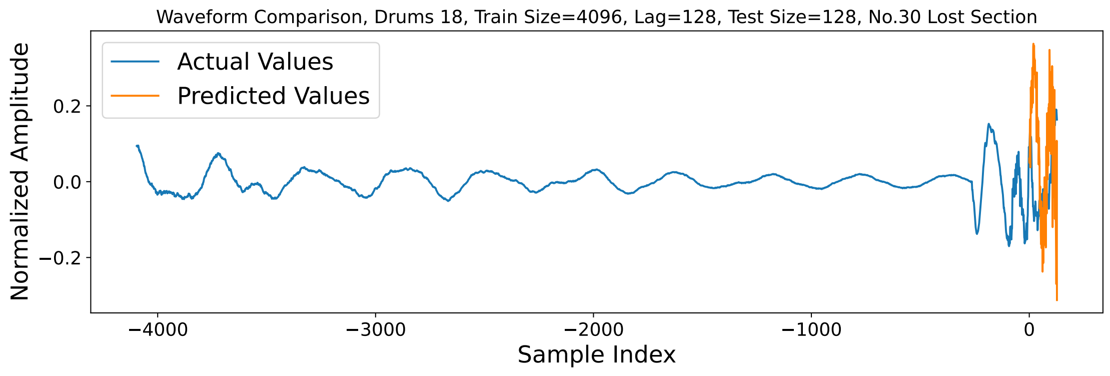

Figure 5.9: A waveform Comparison for Drum ID:18 at the No.30 lost section

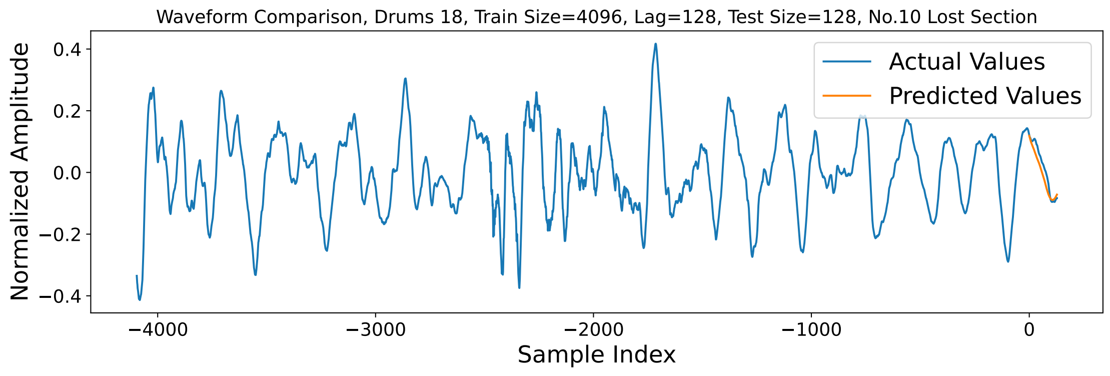

Figure 5.10: A waveform Comparison for Drum ID:18 at the No.10 lost section

vocals.

From generic 24 to 27, most of the cases have lower error when lag numbers increase, which is the phenomenon we expected.

For songs from 0 to 5, in most of the cases, the errors of our AR models are located nearby the ones of benchmark 0, which indicates the performances are not so outstanding than benchmarks.

For song 0 in Figure 5.32, the RMSE is going down from around 0.13 to around 0.07 when lag increase from 1 to 128. The confidence intervals have average length around 0.02 which is actually relatively large. After lag 64, the performances are slightly better than benchmark 0.

For song1 in Figure 5.33, song3 inFigure 5.35, song 4 in Figure 5.36 and the song 5 in Figure 5.37, the RMSE are just around the confidence interval of the RMSE of

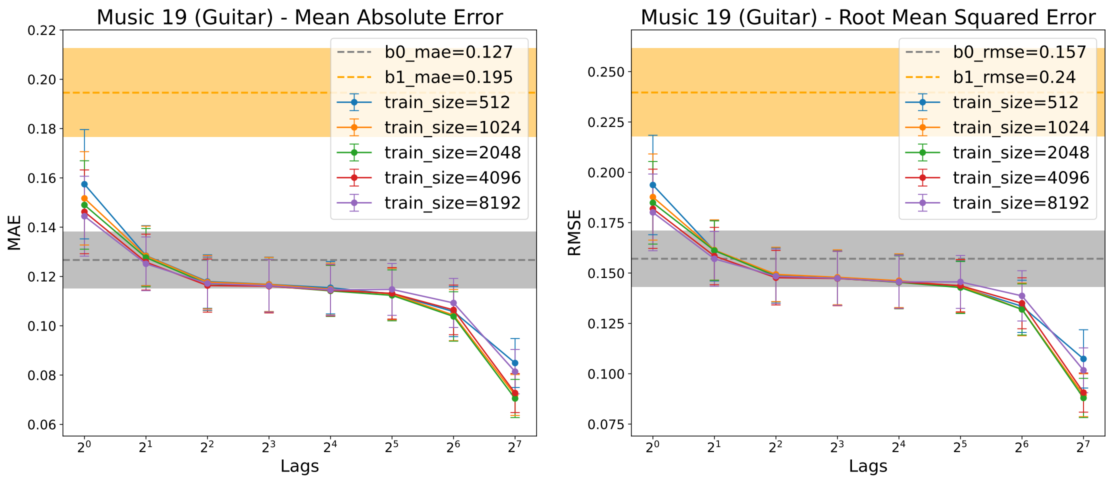

Figure 5.11: Guitar - ID:19 MAE and RMSE

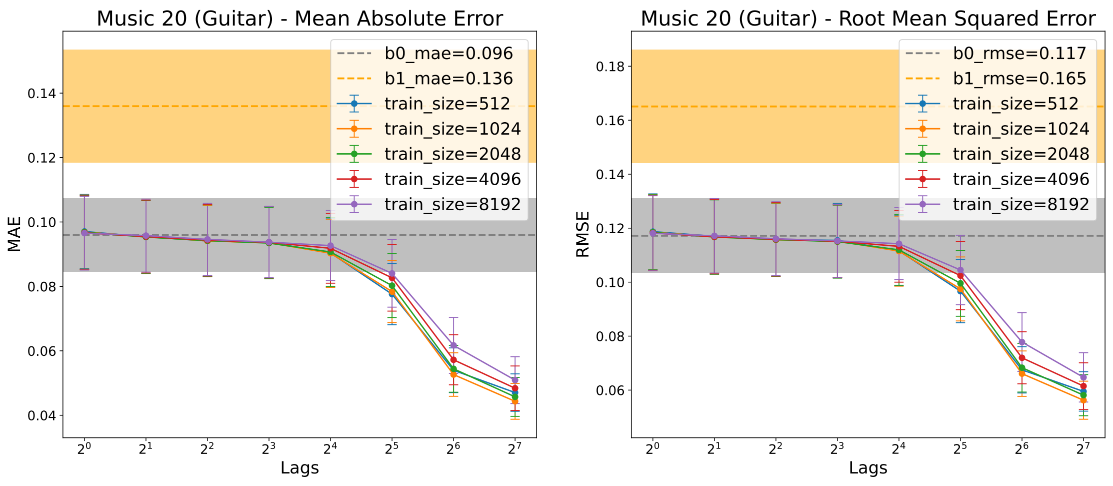

Figure 5.12: Guitar - ID:20 MAE and RMSE

benchmark 0 which means the AR models are not so outstanding. And their minimum RMSE are relatively high with the values around 0.18 and 0.08. Not only they have large confidence interval, but also their trends are irregular.

For song 2 in Figure 5.34, except the models with training size 512 performed worse after lag 32, the trend of RMSE of other models are acceptable and slightly better than benchmarks.

Best pairings of training size and lag number for every music files are shown in Table 5.3. We mainly focus on analysing the effects of changing parameters on the

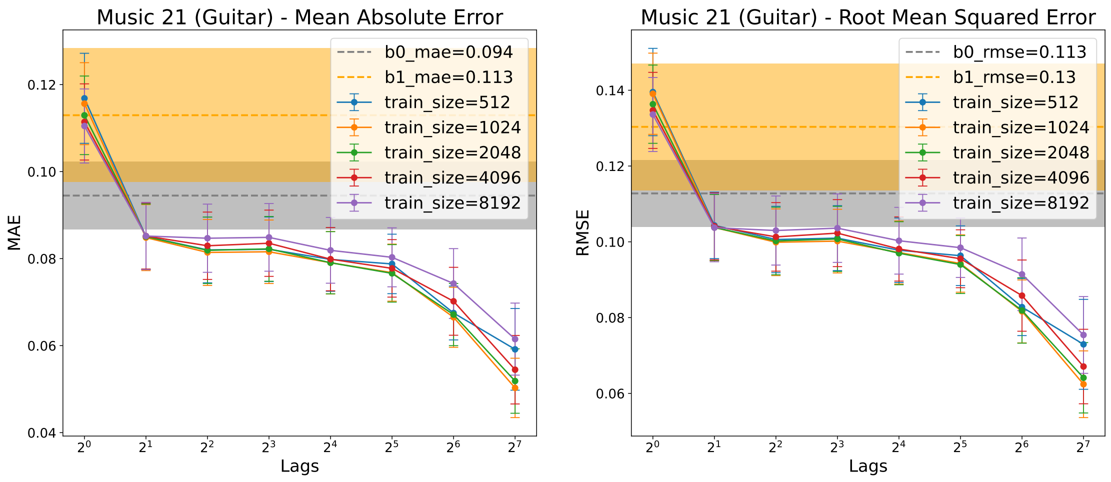

Figure 5.13: Guitar - ID:21 MAE and RMSE

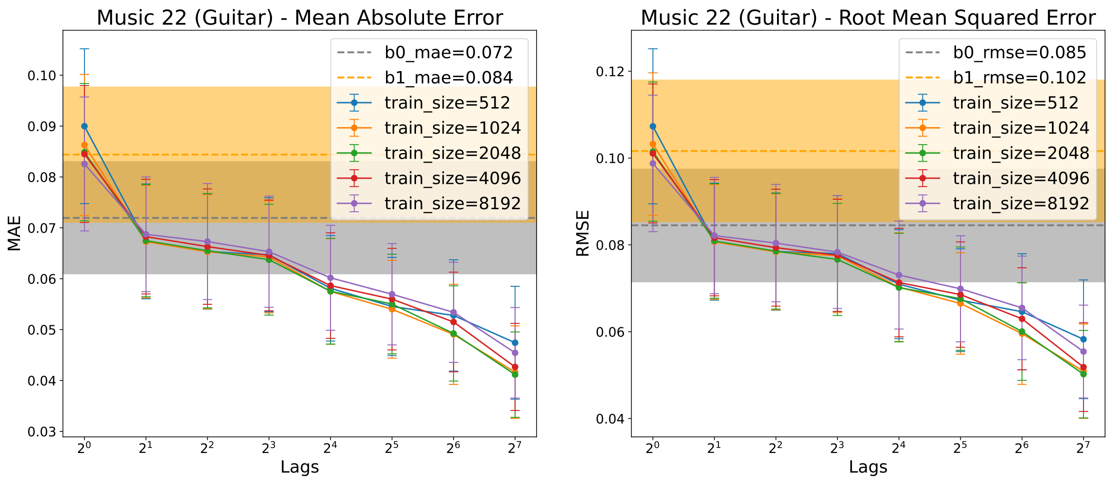

Figure 5.14: Guitar - ID:22 MAE and RMSE

prediction results. For most of the cases, lag 128 is a best choice, and a relatively small training size is enough, e.g. 1024 or 2048. The same results of pairing are found regardless of performance metrics. In all the cases, AR models are showing a better performance than that of two benchmarks, by checking the third, seventh and eighth column for MAE, and the fifth, ninth and tenth column for RMSE at any row in Table 5.3.

What's more, we need to point out that the results of the case with testing size 64 are almost the same with that of the case with testing size 128.

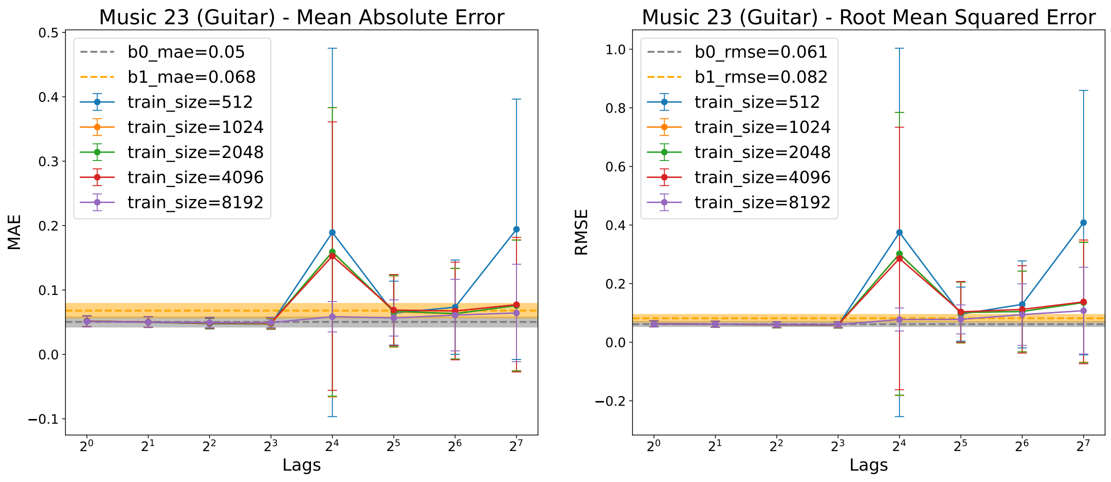

Figure 5.15: Guitar - ID:23 MAE and RMSE

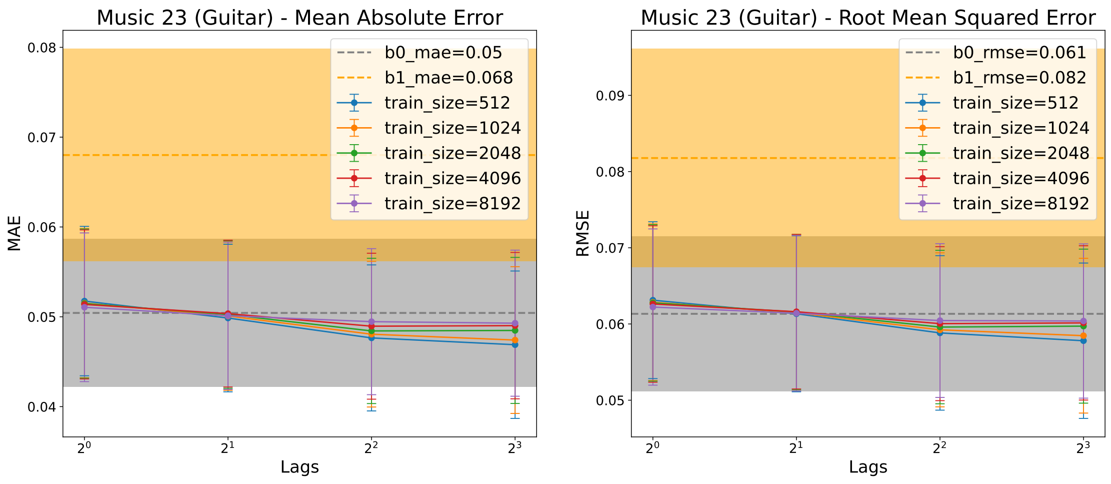

Figure 5.16: Zoomed-In Plot of Guitar - ID:23 MAE and RMSE

5.3 Discussion

The waveform plots of four drum music files are in Figure 5.38 for Drum 15, in Figure 5.39 for Drum 16, in Figure 5.40 for Drum 17 and in Figure 5.41 for Drum 18. From the patterns of those waveform we can see that, in the drum music files, the variances of the normalized amplitudes of the music samples are relatively large e.g. from 0 to 1. The dynamics of Drum 15, 17 and 18 are volatile and the patterns look irregular while Drum 16, which has a slightly better performance, has relatively stable waveform pattern. It explains why our AR models have poor

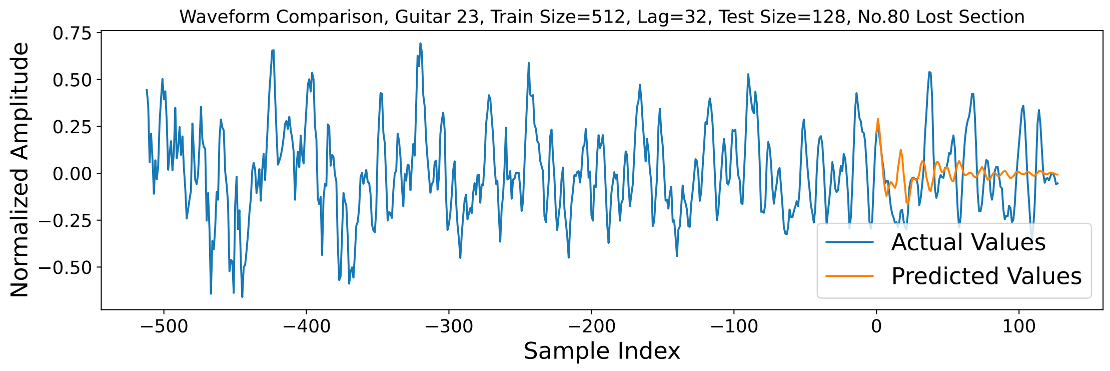

Figure 5.17: A waveform Comparison for Guitar ID:23 at the No.80 lost section

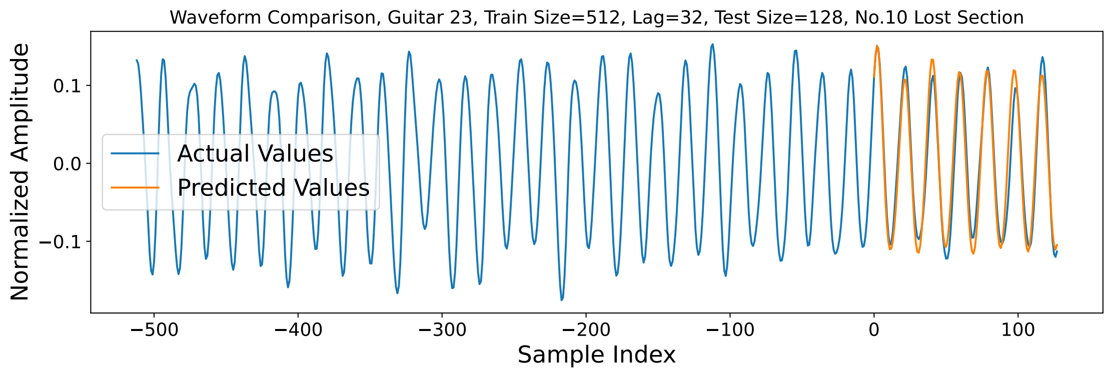

Figure 5.18: A waveform Comparison for Guitar ID:23 at the No.10 lost section

performance on predicting the music data with volatile rhythm. When the data waveform is inconsistent, the far history data may relate low with the value that is being predicted. As a consequence, our AR model may perform worse or uncertainly when we increase the lag numbers.

From the results we got, we may say that our AR models are outperforming benchmarks on forecasting for music sometimes, especially for piano and violin music files. Examples of comparison between the predicted values and the actual values for both are shown in Figure 5.42 and Figure 5.43. When the waveform patterns look more calm and stable, we may have a better performance of AR models.

In Figure 5.44 and Figure 5.45 we can see the performance is just acceptable but worse than the ones we got in the cases for piano music.

The waveform comparisons in Figure 5.8 and Figure 5.48 are implying our AR

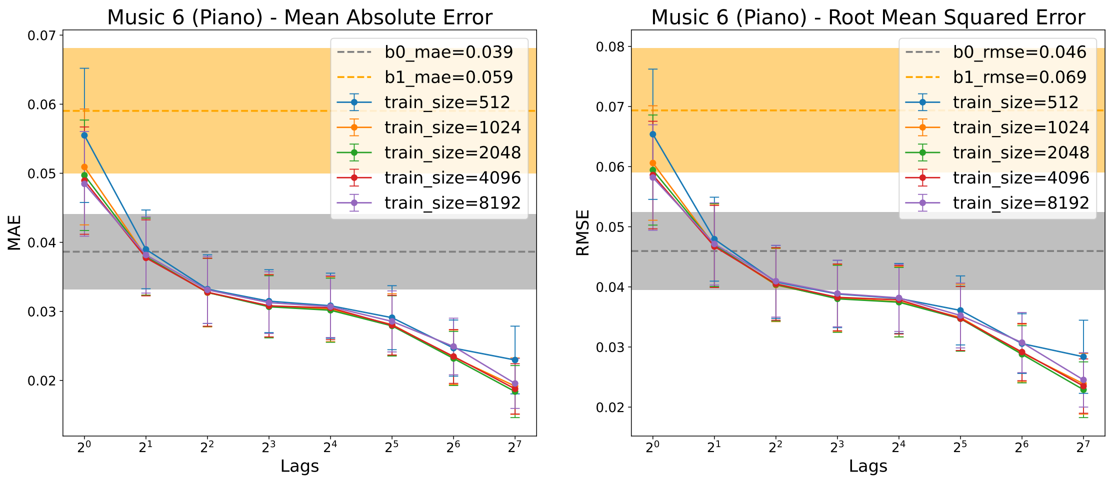

Figure 5.19: Piano - ID:6 MAE and RMSE

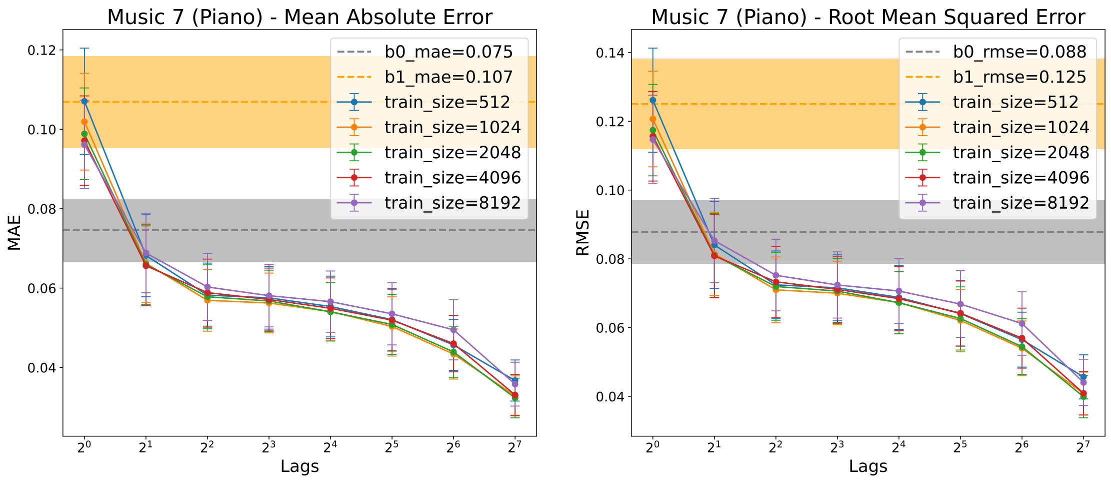

Figure 5.20: Piano - ID:7 MAE and RMSE

models have poor performance on predicting the music data with volatile rhythm, e.g. drums and general mixture songs.

From all the MAE and RMSE plots shown above and in Table 5.3, We may say AR models can reduce the errors in the music data. For w.r.t the benchmarks relatively calm and gentle music files, e.g. piano 9, shown in Figure 5.50 the training size 512 or 1024 and lag 128 would be enough to train the models, and it will bring a large time overhead, e.g. 40-50 ms, when the training size goes beyond 4096. In terms of time consumption, the training time is proportional to the training size and lag number, but when a lag number is fixed, the training size

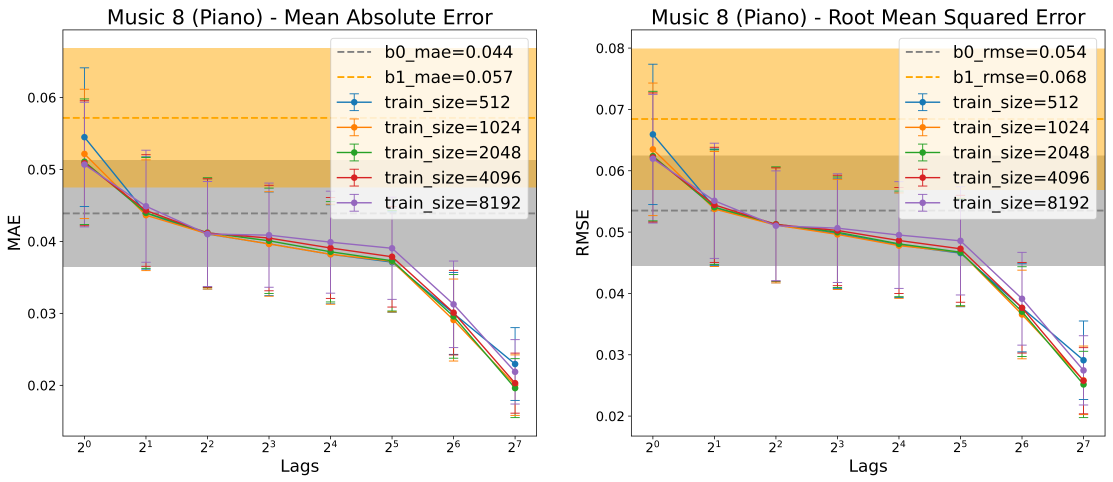

Figure 5.21: Piano - ID:8 MAE and RMSE

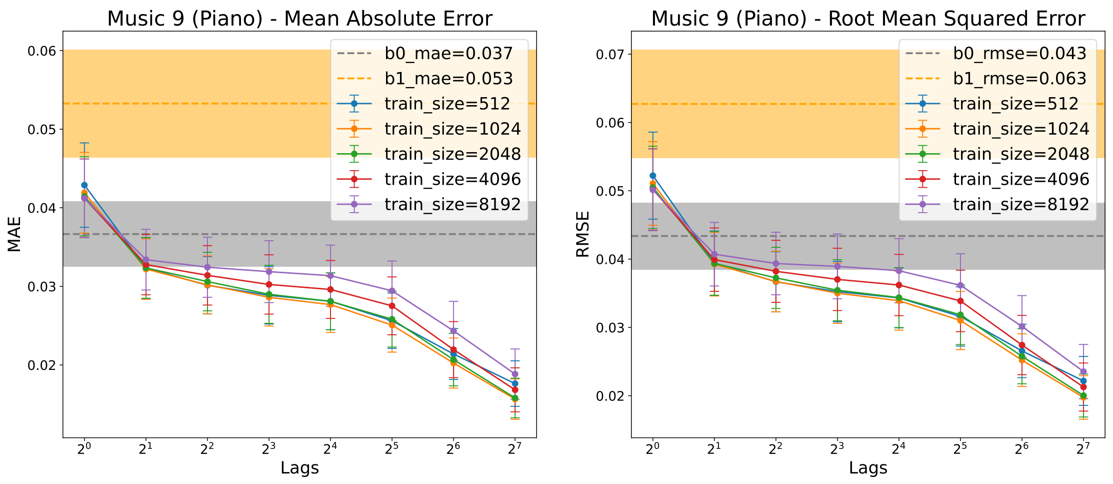

Figure 5.22: Piano - ID:9 MAE and RMSE

does not affect much the prediction time consumption, also shown in Figure 5.50.

However, for the music data with fast rhythm like drums or general songs, lag 128 seems not enough. For these 28 music files, our AR models for predicting piano music have better performances than benchmarks and the other models for other genres. The reason why we get better performance in piano cases may be that the waveform patterns are more regular in piano music, as shown in Figure 5.42.

And we figure out that no matter what kinds of instrument is playing in the music data, or it is only melodies or vocals, as long as we find the regular patterns in its

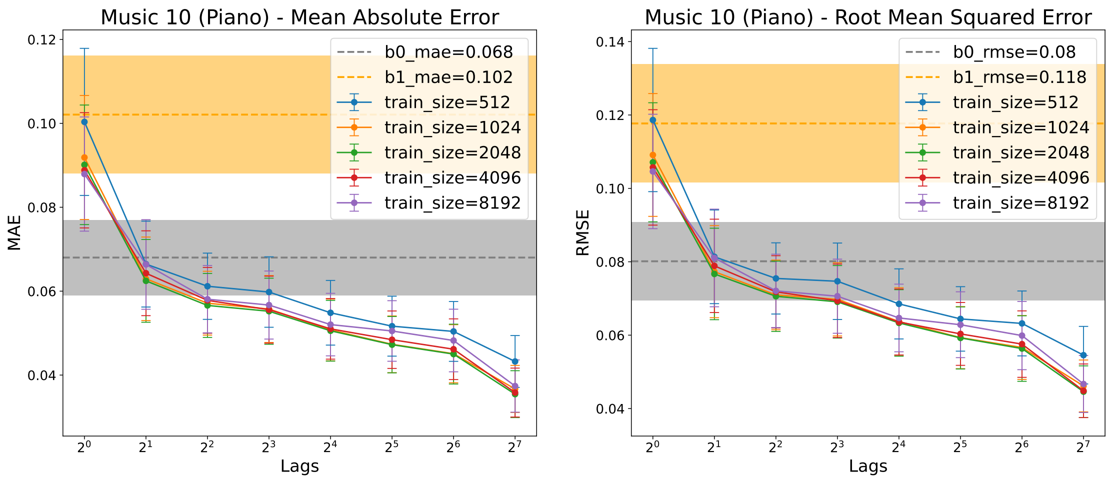

Figure 5.23: Piano - ID:10 MAE and RMSE

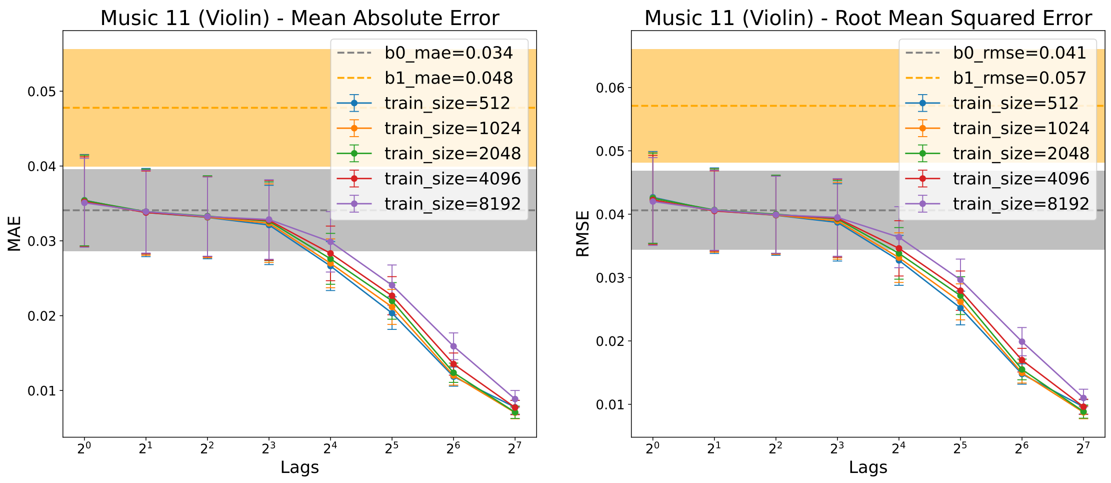

Figure 5.24: Violin - ID:11 MAE and RMSE

waveform, the predicted values are quite close to the actual ones, as observed in Figure 5.42 for piano music, Figure 5.44 for guitar music and Figure 5.45, Figure 5.46, Figure 5.47 for generic audio music and Figure 5.49 for generic mixture songs. Therefore, we may say the instruments do not strongly affect the performance of AR models, but the other parameters do, e.g. the training size and lag values, etc.

For most of the cases here, especially for music data with regular waveform patterns, increasing the lag number, i.e. the order of the model to be trained, generally leads to a better performance, However, increasing the training size of

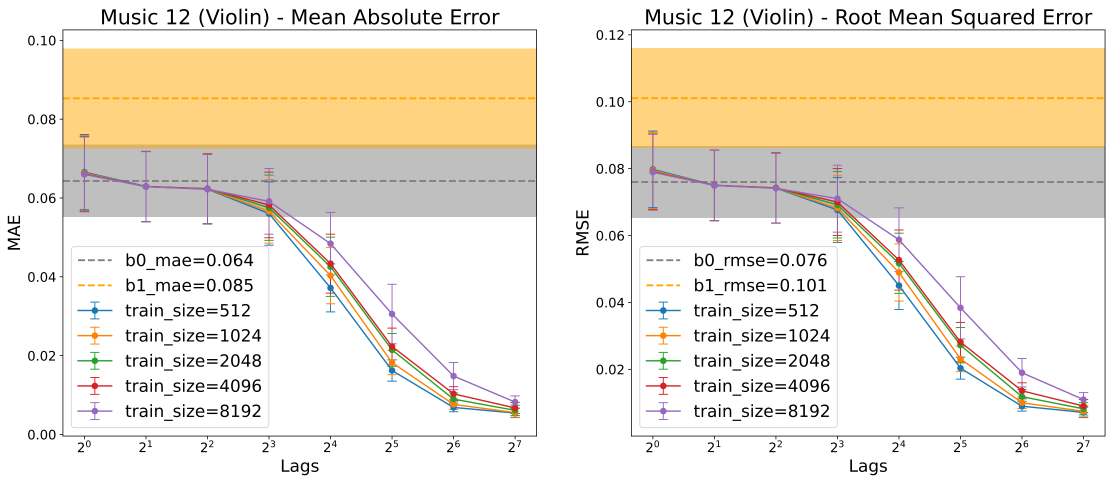

Figure 5.25: Violin - ID:12 MAE and RMSE

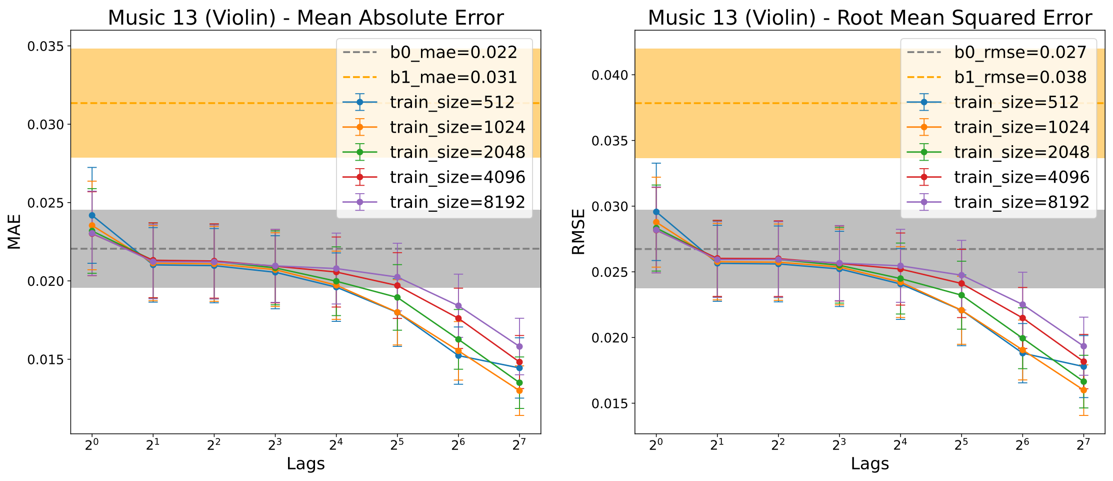

Figure 5.26: Violin - ID:13 MAE and RMSE

the models does not always bring a better performance, and we may introduce much more time overhead for training and predicting. Therefore, we need to pick an appropriate training size along with a trade-off with time consumption and pick a relatively large lag number to have a better solution. Once the training size of the model is defined, a limit is set for the lag number. Thus, if the training size is too small, the lag number may be not enough to train an effective AR model for the complex music data, e.g. drum or general song.

The results so far we got are showing that there are limitations on predicting irregular or volatile music data by AR models. The specific values of the

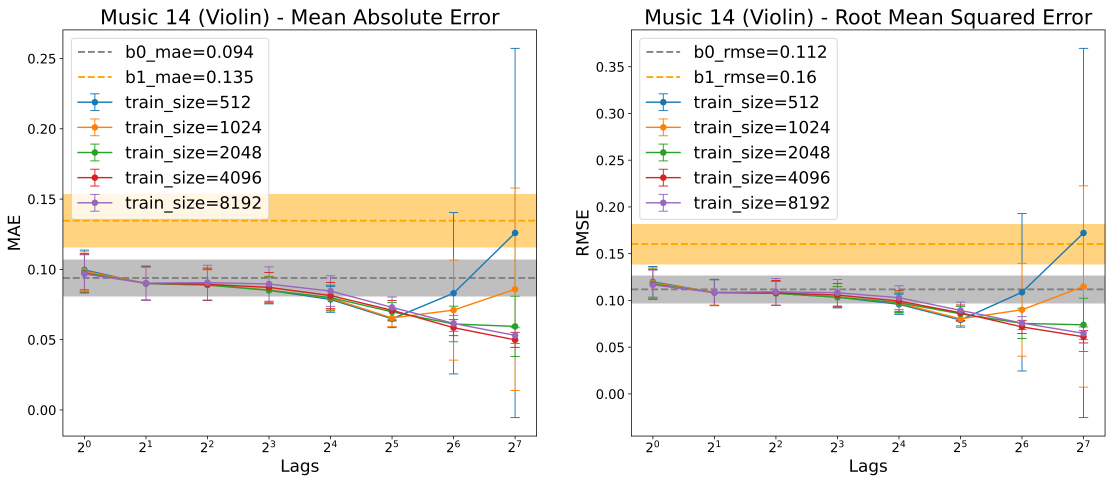

Figure 5.27: Violin - ID:14 MAE and RMSE

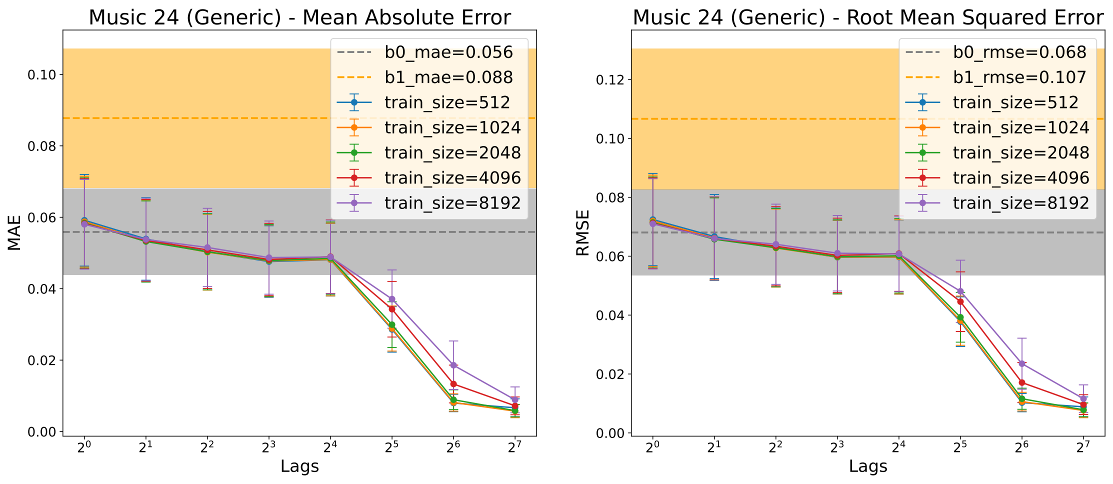

Figure 5.28: Generic - ID:24 MAE and RMSE

parameters we discuss in this thesis are not enough for the complex music data with irregular waveform patterns, e.g. drum or general song.

Despite that our AR models have their limitations, for most of the cases, our AR models outperform two benchmarks (pattern replication and silence substitution), and even in the worst cases we still have slightly better performances than the average performances of the benchmarks when lag is large.

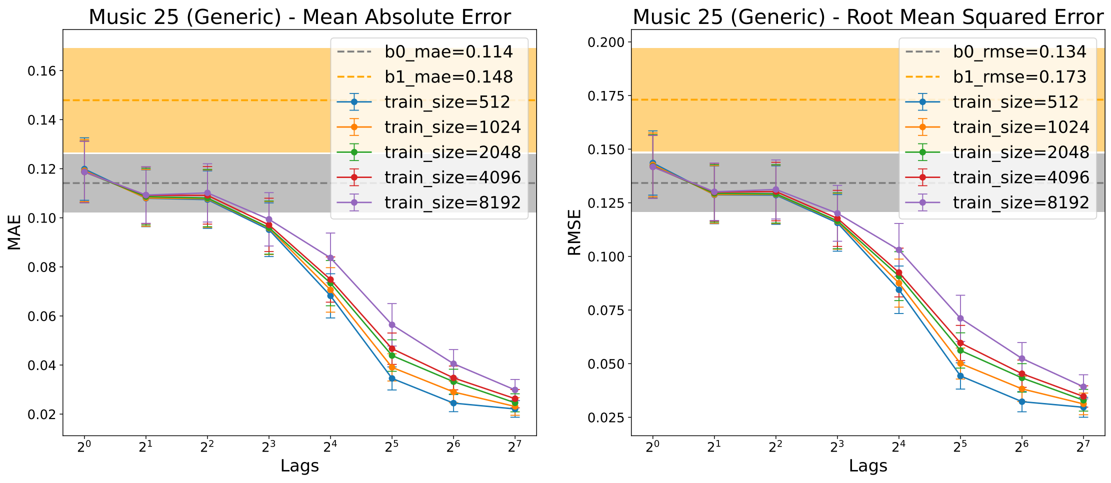

Figure 5.29: Generic - ID:25 MAE and RMSE

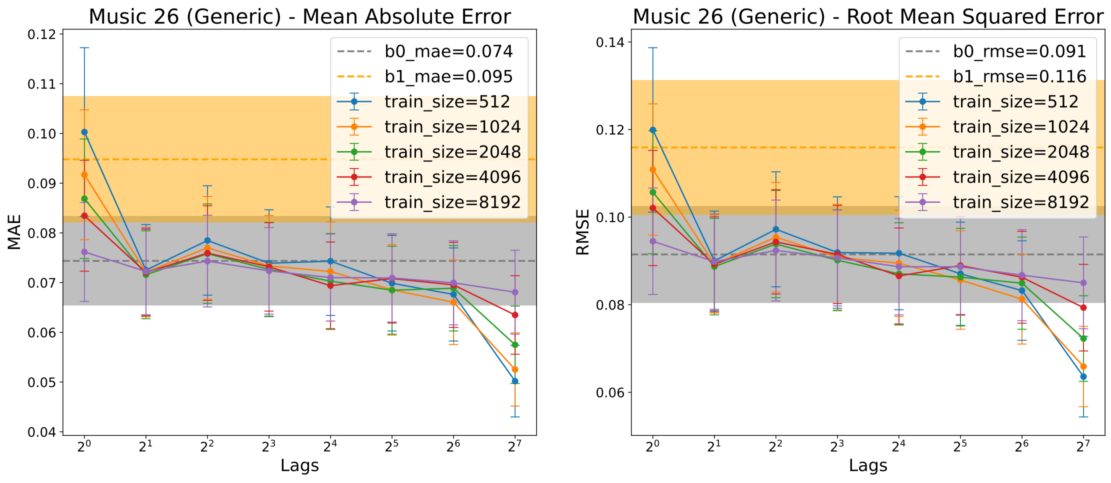

Figure 5.30: Generic - ID:26 MAE and RMSE

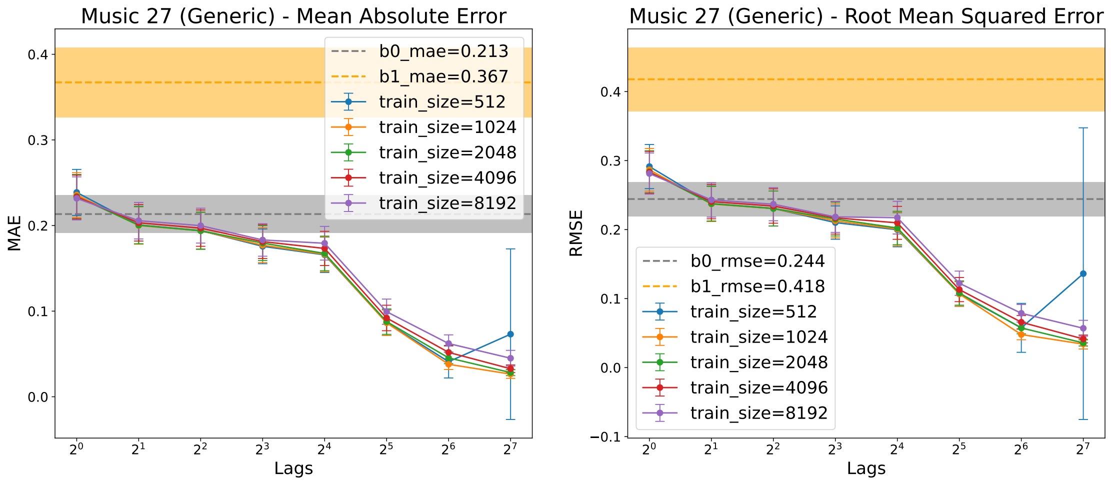

Figure 5.31: Generic - ID:27 MAE and RMSE

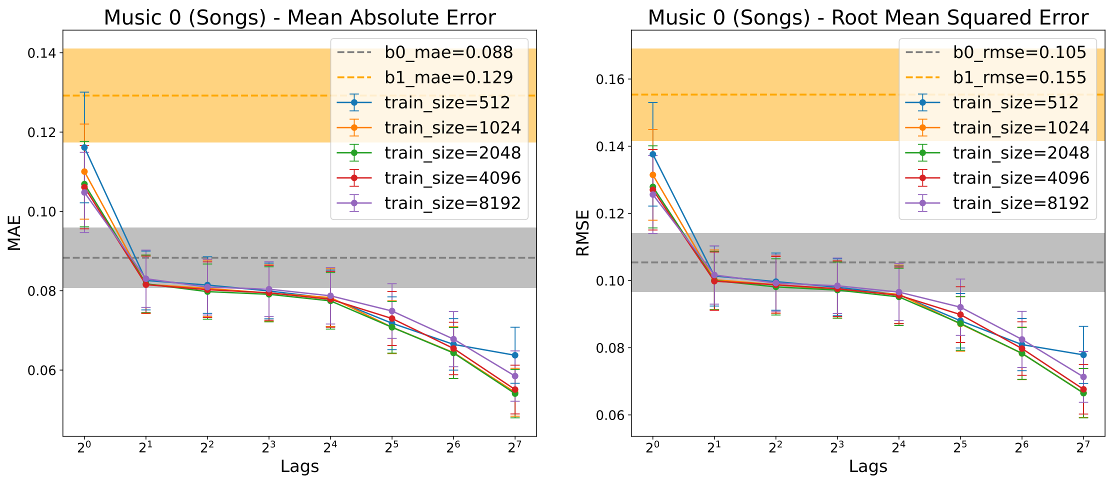

Figure 5.32: Song - ID:0 MAE and RMSE

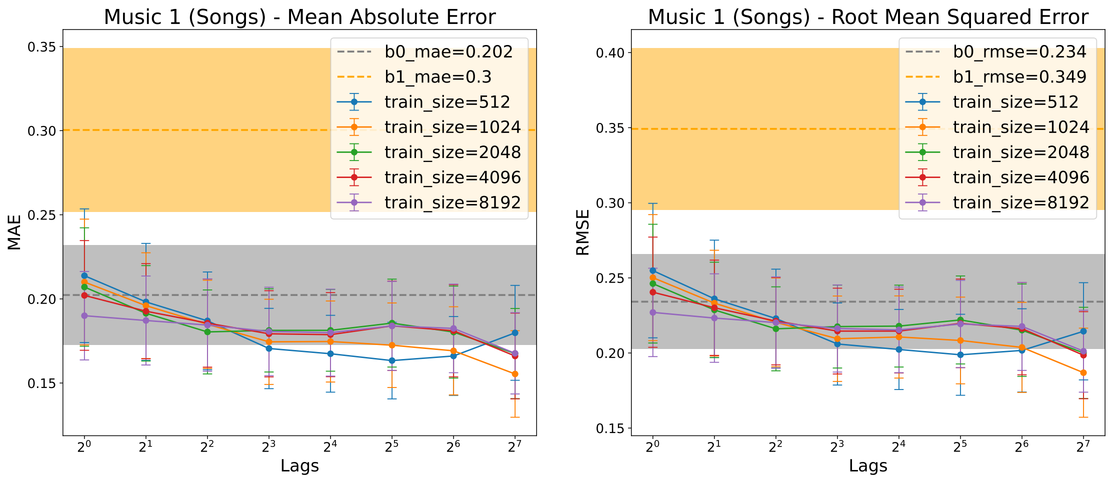

Figure 5.33: Song - ID:1 MAE and RMSE

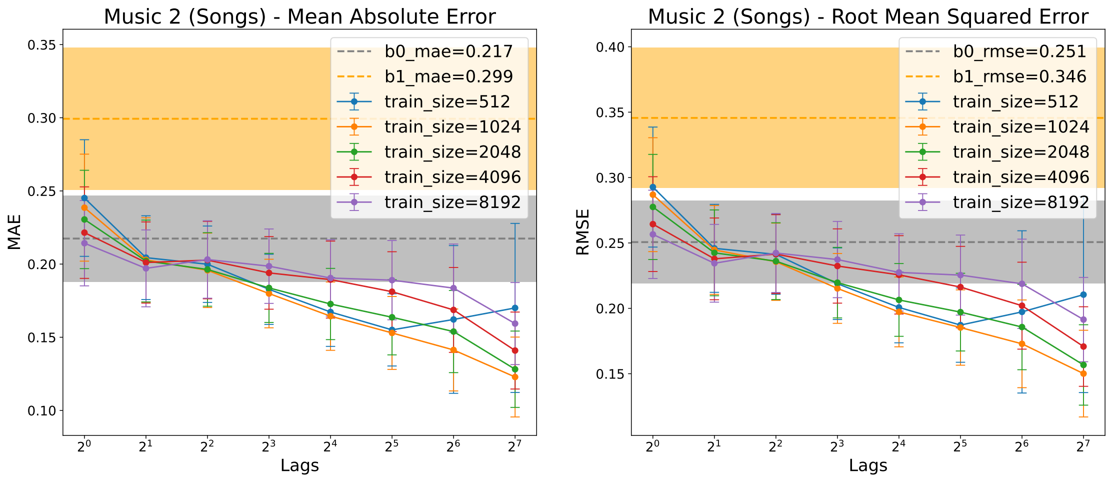

Figure 5.34: Song - ID:2 MAE and RMSE

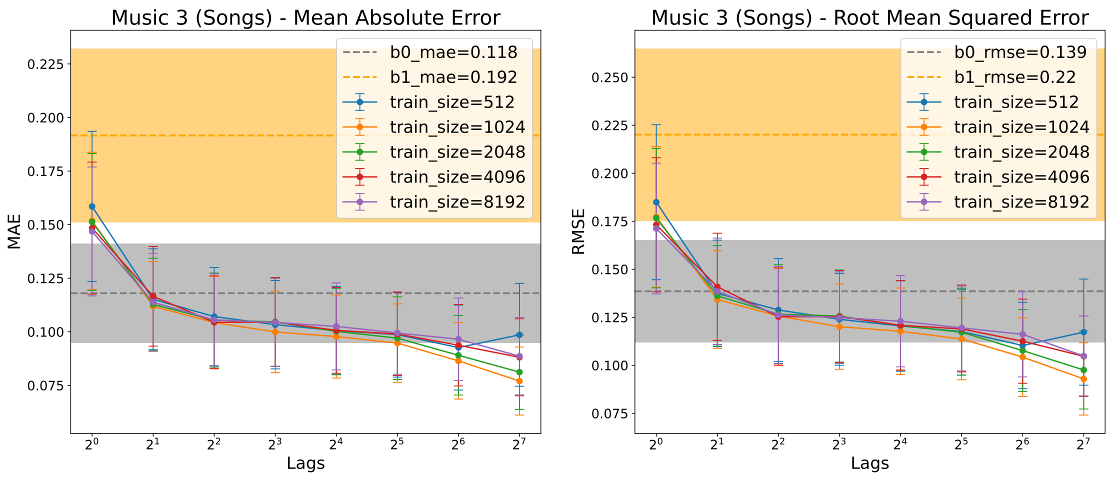

Figure 5.35: Song - ID:3 MAE and RMSE

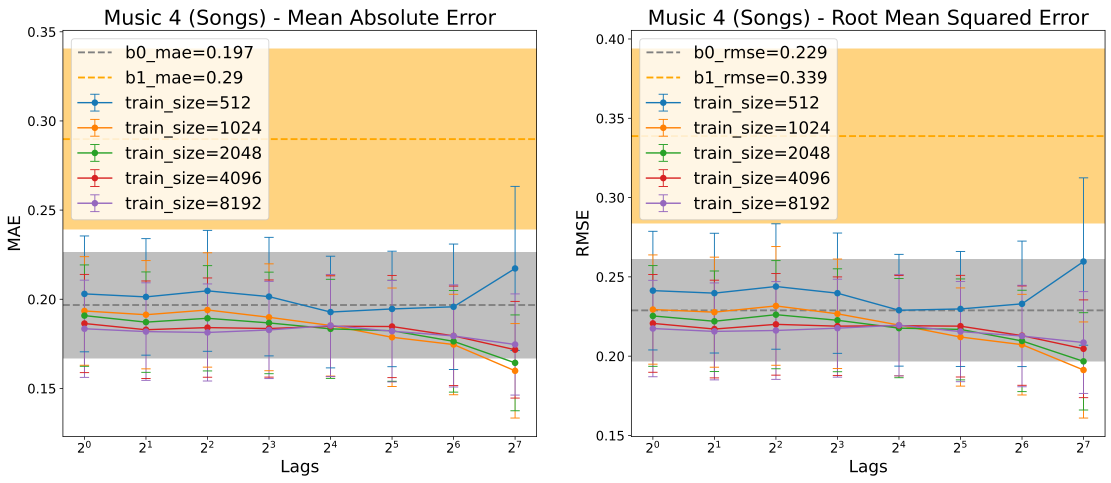

Figure 5.36: Song - ID:4 MAE and RMSE

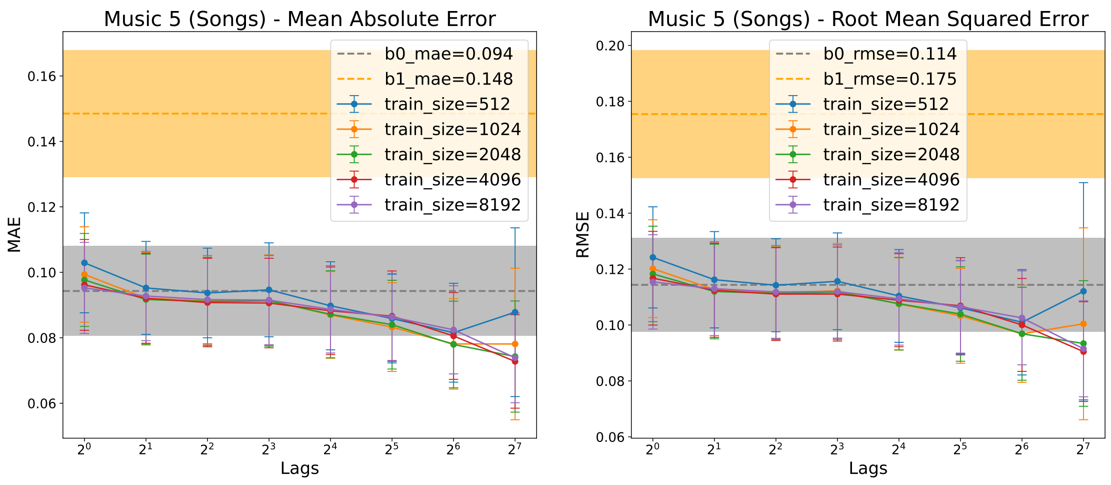

Figure 5.37: Song - ID:5 MAE and RMSE

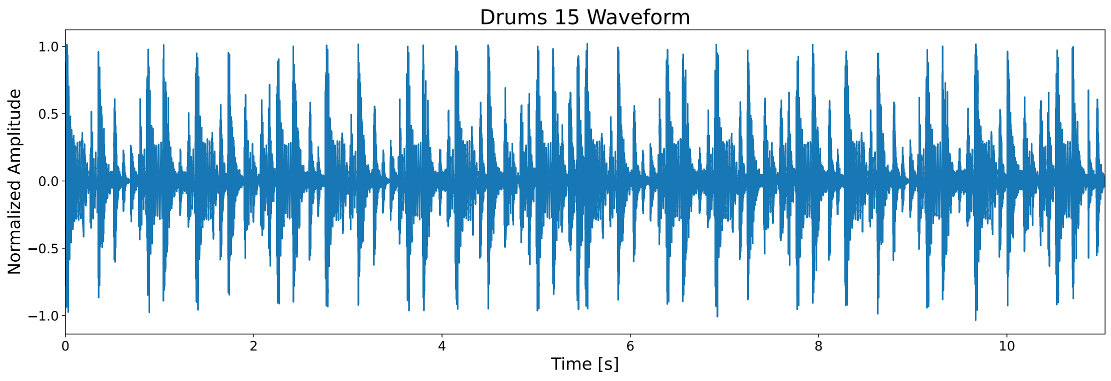

Figure 5.38: The Waveform Overview of Drum ID:15

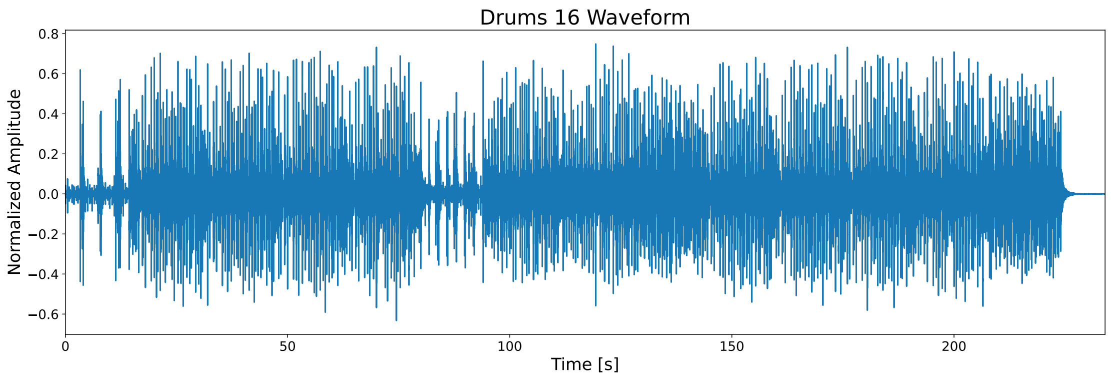

Figure 5.39: The Waveform Overview of Drum ID:16

ID	Genre	Min. MAE	Best Pair (MAE)	Min. RMSE	Best Pair (RMSE)	MAE b0	MAE b1	RMSE b0	RMSE b1
0	Songs	0.054063092	(2048, 128)	0.066480128	(2048, 128)	0.088307095	0.129207581	0.105412471	0.155363621
1	Songs	0.155377889	(1024, 128)	0.186866493	(1024, 128)	0.20230087	0.300391644	0.234138918	0.349200055
2	Songs	0.122838938	(1024, 128)	0.150092782	(1024, 128)	0.217449329	0.29930824	0.250639268	0.345605977
3	Songs	0.077014765	(1024, 128)	0.092904563	(1024, 128)	0.118014493	0.191628501	0.138539572	0.22006845
4	Songs	0.15985491	(1024, 128)	0.191276187	(1024, 128)	0.196762248	0.28980276	0.228871849	0.338766415
5	Songs	0.072768847	(4096, 128)	0.090480626	(4096, 128)	0.094286821	0.14849788	0.114359289	0.175434957
6	Piano	0.018386684	(2048, 128)	0.022893123	(2048, 128)	0.038639758	0.059032973	0.045958797	0.069353472
7	Piano	0.032327406	(2048, 128)	0.039916775	(2048, 128)	0.07457563	0.106905296	0.087820554	0.12503792
8	Piano	0.019612668	(2048, 128)	0.025160385	(2048, 128)	0.043886162	0.057154965	0.053508434	0.068435796
9	Piano	0.015654876	(1024, 128)	0.019744592	(1024, 128)	0.036648681	0.053266931	0.043378582	0.062712597
10	Piano	0.035456018	(2048, 128)	0.044604875	(2048, 128)	0.068013227	0.102080435	0.080134511	0.117695241
11	Violin	0.007041852	(1024, 128)	0.008728786	(1024, 128)	0.034083957	0.04778377	0.040610696	0.057106733
12	Violin	0.005419659	(512, 128)	0.007129611	(512, 128)	0.064314812	0.085283794	0.075995887	0.101062008
13	Violin	0.012994652	(1024, 128)	0.015994493	(1024, 128)	0.022053342	0.031347625	0.026735182	0.037838193
14	Violin	0.049859898	(4096, 128)	0.061090462	(4096, 128)	0.093901605	0.13462241	0.111845822	0.160292337
15	Drums	0.127029089	(512, 16)	0.149644293	(512, 16)	0.151289153	0.225107327	0.174442217	0.261742385
16	Drums	0.032364702	(1024, 128)	0.039759027	(1024, 128)	0.043324412	0.065600753	0.051297342	0.076463376
17	Drums	0.043838238	(512, 32)	0.053742877	(512, 32)	0.051839846	0.077702135	0.061115888	0.09196372
18	Drums	0.035659226	(8192, 16)	0.043050527	(8192, 16)	0.038194218	0.067543194	0.044791903	0.078561812
19	Guitar	0.07049856	(2048, 128)	0.087978379	(2048, 128)	0.126686371	0.194564044	0.157105943	0.239644775
20	Guitar	0.044331195	(1024, 128)	0.056250198	(1024, 128)	0.095933175	0.135936216	0.117198536	0.165092221
21	Guitar	0.050281248	(1024, 128)	0.062416139	(1024, 128)	0.094494491	0.112966053	0.112776553	0.130340226
22	Guitar	0.041152169	(2048, 128)	0.050214748	(2048, 128)	0.071953553	0.084403187	0.084515124	0.101613175
23	Guitar	0.04689295	(512, 8)	0.057802794	(512, 8)	0.050428672	0.068006933	0.061332003	0.081775637
24	Generic	0.005719299	(1024, 128)	0.007622674	(1023, 128)	0.055881729	0.087760337	0.068057666	0.106649136
25	Generic	0.022099709	(512, 128)	0.029668134	(512, 128)	0.114142504	0.14791733	0.134228588	0.173068264
26	Generic	0.050190422	(512, 128)	0.063575763	(512, 128)	0.074367549	0.094800837	0.091485649	0.115910354
27	Generic	0.026468523	(1024, 128)	0.033995412	(1024, 128)	0.213359488	0.367188334	0.244228743	0.417773621

Table 5.3: Best Pairings of training size and lag for each music file

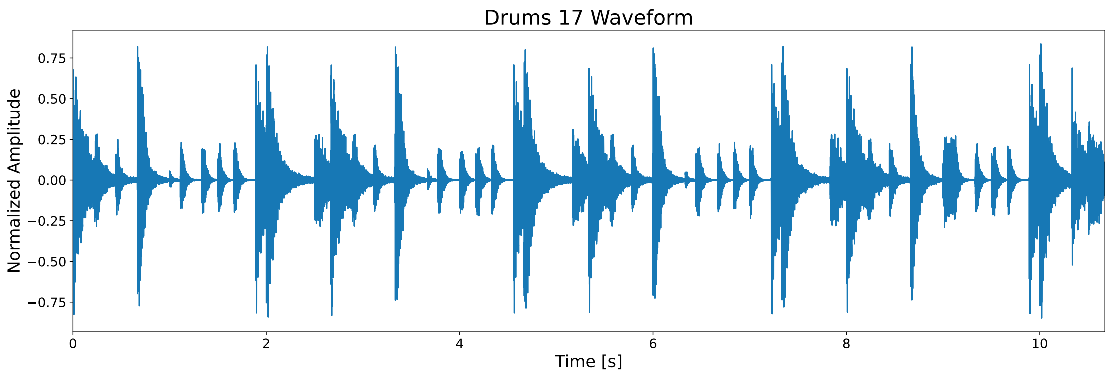

Figure 5.40: The Waveform Overview of Drum ID:17

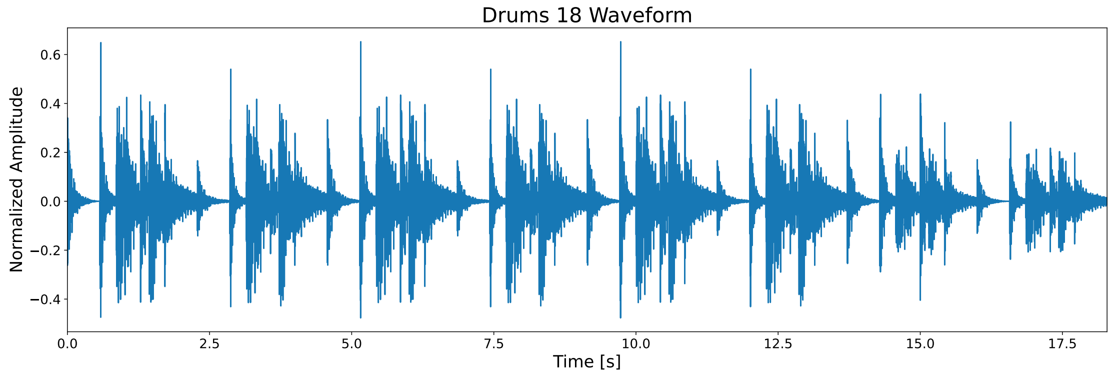

Figure 5.41: The Waveform Overview of Drum ID:18

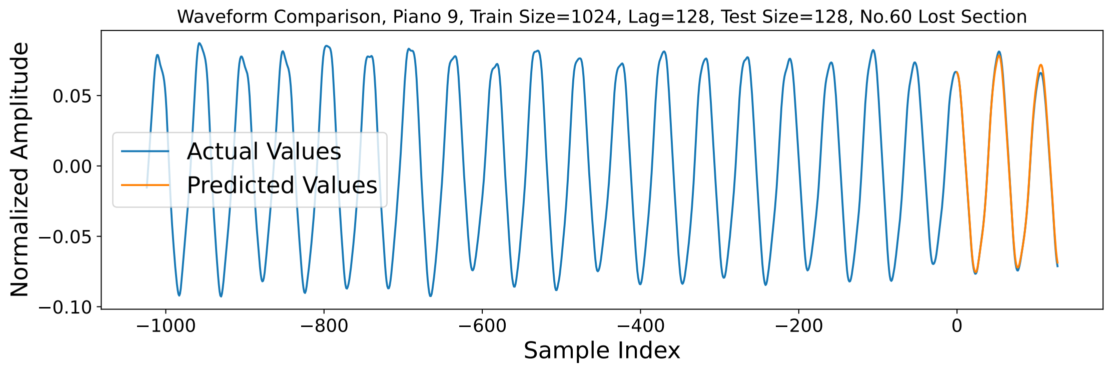

Figure 5.42: A waveform Comparison for Piano ID:9 at the No.60 lost section

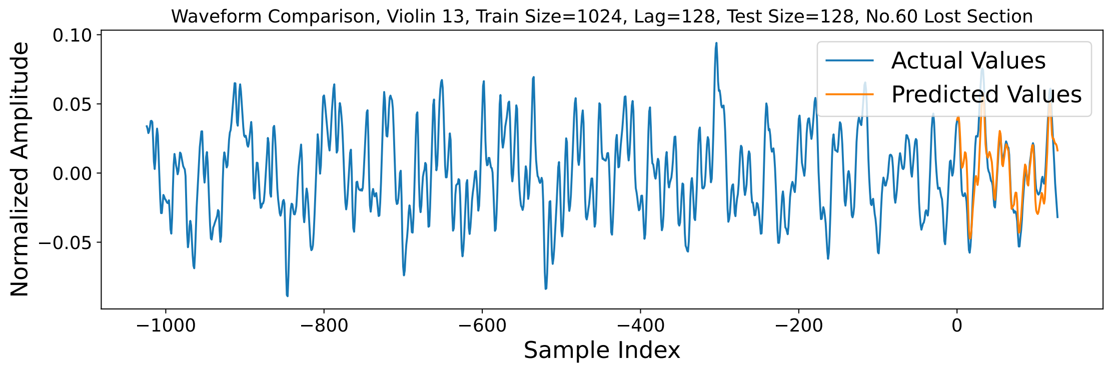

Figure 5.43: A waveform Comparison for Violin ID:13 at the No.60 lost section

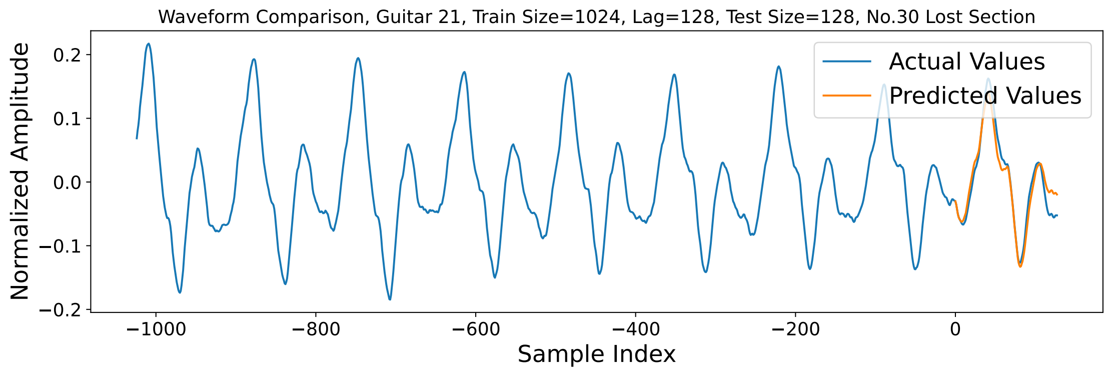

Figure 5.44: A waveform Comparison for Guitar ID:21 at the No.30 lost section

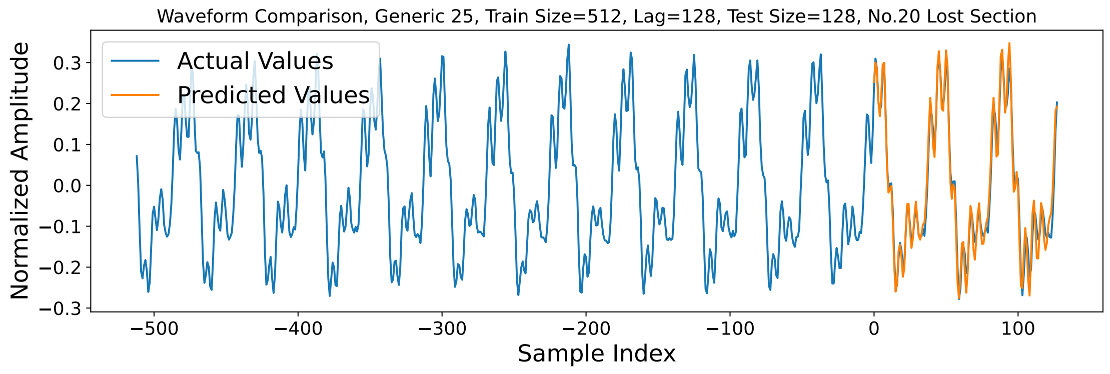

Figure 5.45: A waveform Comparison for Generic ID:25 at the No.20 lost section

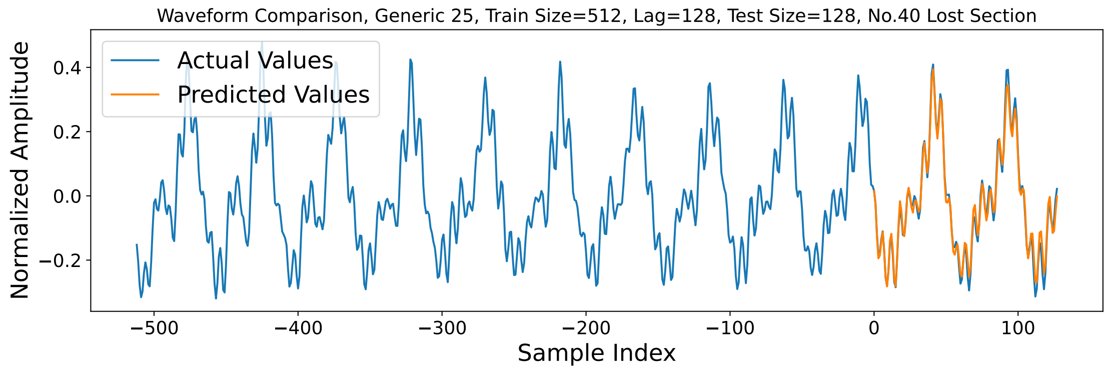

Figure 5.46: A waveform Comparison for Generic ID:25 at the No.40 lost section

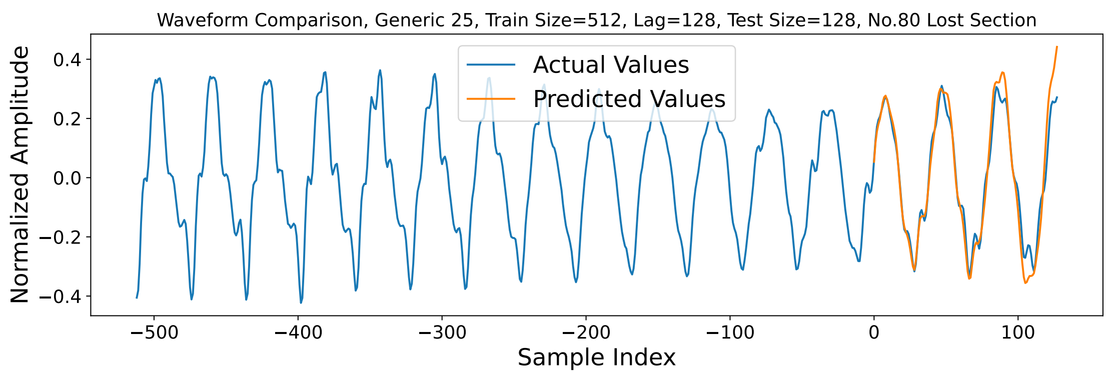

Figure 5.47: A waveform Comparison for Generic ID:25 at the No.80 lost section

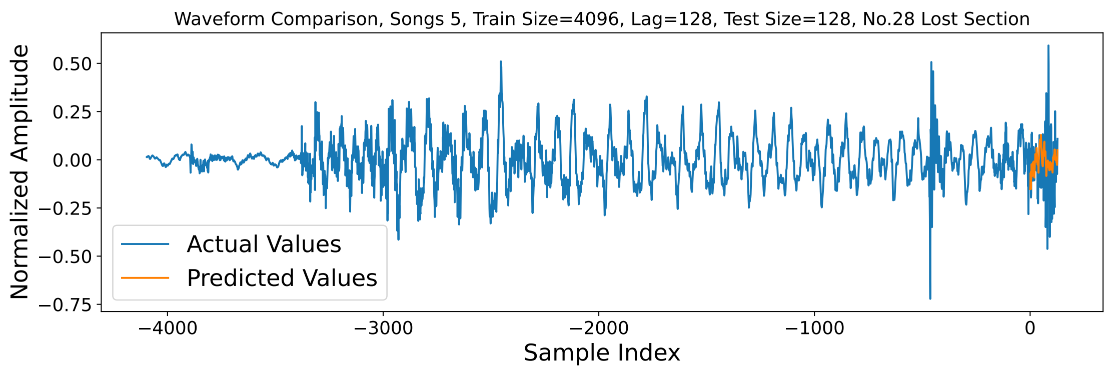

Figure 5.48: A waveform Comparison for Song ID:5 at the No.28 lost section

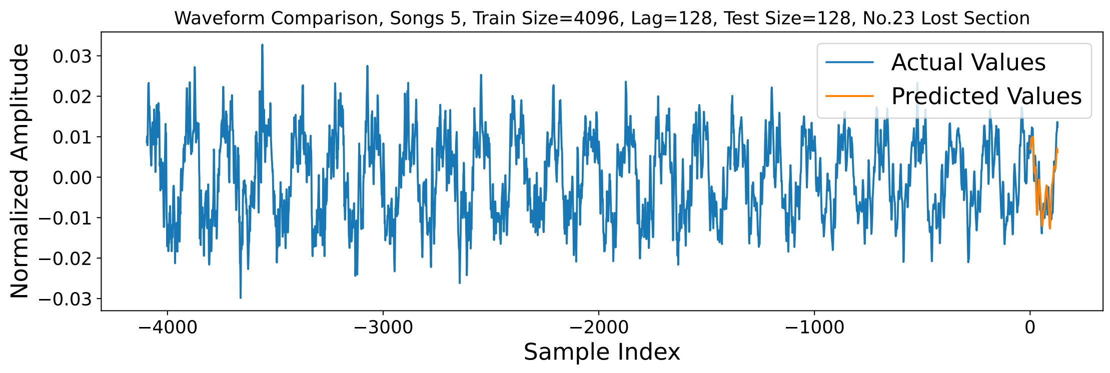

Figure 5.49: A waveform Comparison for Song ID:5 at the No.23 lost section

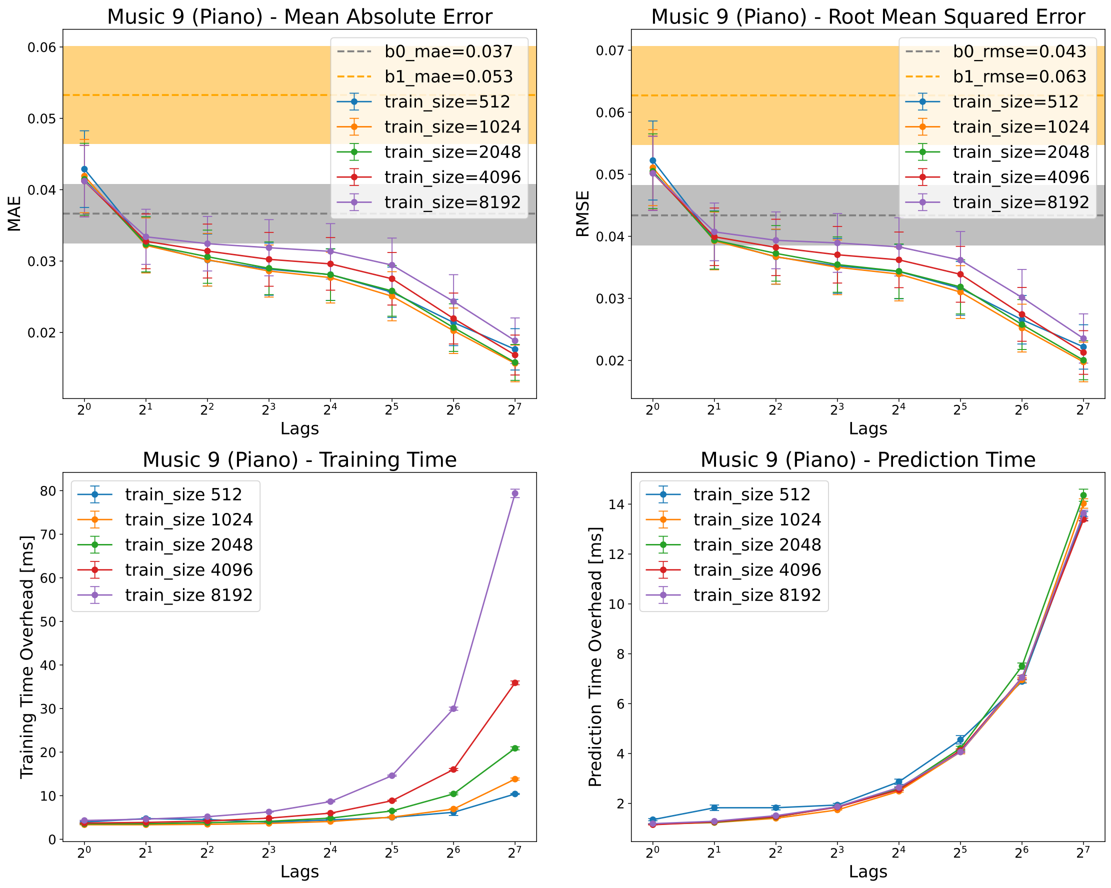

Figure 5.50: Error vs Time Overhead for Piano ID:9

Chapter 6

Conclusion

This thesis describes an Auto-Regressive Model technique for packet loss concealment that is designed to be implemented in real-time networked music performance applications. The experiments are mainly focused on exploring and evaluating the performance of AR models for music data forecasting, by inserting gaps in random positions to mock data losses and then training the models to predict the lost data section. And we may say, under some specific conditions, AR models can reduce the glitches and artifacts due to excessive data delays or data losses in audio streams.

For relatively calm and gentle music files or the music samples with regular patterns, the training size 512 or 1024 and lag 128 would be enough to train the models, and it will bring a large time overhead when the training size goes beyond 4096. However, for music data with relatively fast rhythm like drums or general songs, lag 128 seems not enough.

In our data set of 28 music files, our AR models for predicting piano music outperform the benchmarks as well as the AR models for other music files. The reason why we get better performance in piano cases is that we found more regular waveform patterns in piano music samples. And we figure out that no matter what kinds of instrument is producing the music data, and no matter it is only melodies or vocals, as long as we find the regular patterns in its waveform, the predicted values are quite close to the actual ones. Therefore, we may say the instruments do not significantly affect the performance of PLC based on AR models, but the other parameters do, e.g. training size and lag values, etc.

For most of the cases here, especially for the music data with regular waveform patterns, increasing the lag number, i.e. the order of the model to be trained, generally leads to a better performance, However, increasing the training size does not gain much in terms of prediction performance, by which we may introduce much more time overhead for training and predicting. And by defining the training size, we also set a limit for the lag number, so if the training size is too small, the lag number may be not enough to train an effective AR model for complex music data. Thus, we need to pick an appropriate training size along with a trade-off with time consumption and pick a lag number as large as possible to have a better solution. In terms of time consumption, the training time is proportional to the training size and lag number, but when a lag number is fixed, the training size does not affect much on the prediction time consumption.

The results so far we got are showing that AR models have limitations on predicting irregular or volatile music data. The specific values of the parameters we discuss in this thesis are not enough for the complex music data with irregular waveform patterns. And we may find discontinuities at the edges which connect the predicted waveform and the actual one after we synthesize the samples and insert them back into the audio streams. And finding specific techniques to solve this problem could be an interesting future work.

Despite that our AR models have their limitations, for most of the cases, our AR models outperform two benchmarks (pattern replication and silence substitution), and even in the worst cases we still have slightly better performances than the average performances of the benchmarks when lag is large.

Exploring packet loss concealment approaches (PLC) of NMP is becoming a more and more interesting topic thanks to the rapid growth of the Internet and increasing international or inter-regional communications. Nowadays, artificial intelligence technologies are rapidly developing, such as machine learning and deep learning, so predicting the lost music data in real-time audio streams by using AI models are likely possible. For future work, the results of AR model prediction on lost music data in this thesis would be useful as a comparison for future PLC techniques.

Bibliography

[1] Prateek Verma, Alessandro Ilic Mezza, Chris Chafe, and Cristina Rottondi. «A deep learning approach for low-latency packet loss concealment of audio signals in networked music performance applications». In: *2020 27th Conference of Open Innovations Association (FRUCT)*. IEEE. 2020, pp. 268–275 (cit. on pp. 1, 7).

[2] Alexander Carôt, Pedro Rebelo, and Alain Renaud. «Networked Music Performance: State of the Art». In: 2007 (cit. on p. 4).

[3] *Indigenous to the Net: Early Network Music Bands in the San Francisco Bay Area.* `http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html`.. Last accessed: 08 February, 2022 (cit. on p. 5).

[4] Chris Chafe, Scott Wilson, Randal Leistikow, Dave Chisholm, and Gary Scavone. «A simplified approach to high quality music and sound over IP». In: *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*. Citeseer. 2000, pp. 159–164 (cit. on p. 5).

[5] Alexander A Sawchuk, Elaine Chew, Roger Zimmermann, Christos Papadopoulos, and Chris Kyriakakis. «From remote media immersion to distributed immersive performance». In: *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*. 2003, pp. 110–120 (cit. on pp. 5, 6).

[6] Juan-Pablo Cáceres and Chris Chafe. «JackTrip: Under the hood of an engine for network audio». In: *Journal of New Music Research* 39.3 (2010), pp. 183–187 (cit. on p. 5).

[7] Alexander Carôt and Christian Werner. «Distributed network music workshop with soundjack». In: *Proceedings of the 25th Tonmeistertagung, Leipzig, Germany* (2008) (cit. on p. 5).

[8] Chrisoula Alexandraki, Panayotis Koutlemanis, Petros Gasteratos, Nikolas Valsamakis, Demosthenes Akoumianakis, Giannis Milolidakis, G Vellis, and D Kotsalis. «Towards the implementation of a generic platform for networked music performance: the diamouses approach». In: *ICMC*. 2008, pp. 251–258 (cit. on p. 5).

[9] Randy Erfa Saputra and Ary Setijadi Prihatmanto. «Design and implementation of beatme as a networked music performance (nmp) system». In: *2012 International Conference on System Engineering and Technology (ICSET)*. IEEE. 2012, pp. 1–6 (cit. on p. 5).

[10] Xiaoyuan Gu, Matthias Dick, Zefir Kurtisi, Ulf Noyer, and Lars Wolf. «Network-centric music performance: Practice and experiments». In: *IEEE Communications Magazine* 43.6 (2005), pp. 86–93 (cit. on p. 5).

[11] Robin Renwick. «Sourcenode: a network sourced approach to network music performance (nmp)». In: *ICMC*. 2012 (cit. on p. 5).

[12] Charilaos Stais, Yannis Thomas, George Xylomenos, and Christos Tsilopoulos. «Networked music performance over information-centric networks». In: *2013 IEEE International Conference on Communications Workshops (ICC)*. IEEE. 2013, pp. 647–651 (cit. on p. 5).

[13] Chris Chafe. «Living with net lag». In: *Audio Engineering Society Conference: 43rd International Conference: Audio for Wirelessly Networked Personal Devices*. Audio Engineering Society. 2011 (cit. on p. 5).

[14] Alexander Carôt and Christian Werner. «Fundamentals and principles of musical telepresence». In: *Journal of Science and Technology of the Arts* 1.1 (2009), pp. 26–37 (cit. on p. 6).

[15] Cristina Rottondi, Michele Buccoli, Massimiliano Zanoni, Dario Garao, Giacomo Verticale, and Augusto Sarti. «Feature-based analysis of the effects of packet delay on networked musical interactions». In: *Journal of the Audio Engineering Society* 63.11 (2015), pp. 864–875 (cit. on p. 6).

[16] Nelson Posse Lago and Fabio Kon. «The quest for low latency». In: *ICMC*. 2004 (cit. on p. 6).

[17] Teemu Mäki-Patola. «Musical effects of latency». In: *Suomen Musiikintutkijoiden* 9 (2005), pp. 82–85 (cit. on p. 6).

[18] Anders Askenfelt and Erik V Jansson. «From touch to string vibrations. I: Timing in the grand piano action». In: *The Journal of the Acoustical Society of America* 88.1 (1990), pp. 52–63 (cit. on p. 6).

[19] John Lazzaro and John Wawrzynek. «A Case for Network Musical Performance». In: *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video.* NOSSDAV '01. Port Jefferson, New York, USA: Association for Computing Machinery, 2001, pp. 157–166. ISBN: 1581133707. DOI: `10.1145/378344.378367`. URL: `https://doi.org/10.1145/378344.378367` (cit. on p. 6).

[20] Jussi Virolainen and Pauli Laine. *Methods and apparatus for transmitting MIDI data over a lossy communications channel.* US Patent 6,898,729. May 2005 (cit. on p. 7).

[21] Jason Robert Nelson, Allen James Heidorn, and Robert John Cox. *Reliable real-time transmission of musical sound control data over wireless networks.* US Patent 9,601,097. Mar. 2017 (cit. on p. 7).

[22] Guoqiang Zhang and W Bastiaan Kleijn. «Autoregressive model-based speech packet-loss concealment». In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing.* IEEE. 2008, pp. 4797–4800 (cit. on p. 7).

[23] Marco Fink and Udo Zölzer. «Low-Delay Error Concealment with Low Computational Overhead for Audio over IP Applications.» In: *DAFx.* 2014, pp. 309–316 (cit. on p. 7).

[24] Henning Sanneck, Alexander Stenger, K Ben Younes, and Bernd Girod. «A new technique for audio packet loss concealment». In: *Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference.* IEEE. 1996, pp. 48–52 (cit. on p. 7).

[25] *Forecasting: Principles and Practice.* `https://otexts.com/fpp2/regression-intro.html`. Last accessed: 16 January, 2022 (cit. on p. 9).

[26] *Time plot of actual US consumption expenditure and predicted US consumption expenditure.* `https://otexts.com/fpp2/least-squares.html`. Last accessed: 16 January, 2022 (cit. on p. 10).

[27] SAGE Research Methods Datasets. *Learn about Time Series ACF and PACF in SPSS with Data from the USDA Feed Grains Database (1876–2015).* `https://methods.sagepub.com/base/download/DatasetStudentGuide/time-series-acf-pacf-in-us-feedgrains-1876-2015`. Last accessed: 16 January, 2022 (cit. on p. 12).

[28] *Freesound.* `https://freesound.org/`. Last accessed: 20 February, 2022 (cit. on pp. 16, 26).

[29] *Augmented Dickey Fuller Test - Must Read Guide.* https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/. Last accessed: 17 February, 2022 (cit. on p. 16).

[30] *Tutorial: Understanding Regression Error Metrics in Python.* https://www.dataquest.io/blog/understanding-regression-error-metrics/. Last accessed: 17 February, 2022 (cit. on p. 24).

[31] *Source of song <morning-garden-acoustic-chill-15013.mp3>.* https://pixabay.com/music/acoustic-group-morning-garden-acoustic-chill-15013/. Last accessed: 20 February, 2022 (cit. on p. 28).

[32] *Source of song <jazzy-abstract-beat-11254.mp3>.* https://pixabay.com/music/beats-jazzy-abstract-beat-11254/. Last accessed: 20 February, 2022 (cit. on p. 28).

[33] *Source of song <sexy-fashion-beats-11176.mp3>.* https://pixabay.com/music/beats-sexy-fashion-beats-simulate-11176/. Last accessed: 20 February, 2022 (cit. on p. 28).

[34] *Source of song <madirfan-unreleased-demo-15-01-2022-14254.mp3>.* https://pixabay.com/music/beautiful-plays-madirfan-unreleased-demo-15-01-2022-14254/. Last accessed: 20 February, 2022 (cit. on p. 28).

[35] *Source of song <commercial-rock-beats-11249.mp3>.* https://pixabay.com/music/beats-commercial-rock-beats-spin-11249/. Last accessed: 20 February, 2022 (cit. on p. 28).

[36] *Source of song <bensound-ukulele.mp3>.* https://www.bensound.com/royalty-free-music/track/ukulele. Last accessed: 20 February, 2022 (cit. on p. 28).