Using Autoregressive Models for Real-Time Packet Loss Concealment in Networked Music Performance Applications

Matteo Sacchetto, Yuen Huang, Andrea Bianco, Cristina Rottondi*
{matteo.sacchetto,andrea.bianco,cristina.rottondi}@polito.it
yuen.huang@studenti.polito.it
Politecnico di Torino, Dept. of Electronics and Telecommunications
Turin, Italy

ABSTRACT

In Networked Music Performances (NMP), concealing the effects of lost/late packets on the quality of the playback audio stream is of pivotal importance to mitigate the impact of the resulting audio artifacts. Traditional packet loss concealment techniques implemented in standard audio codecs can be leveraged only at the price of an increased mouth-to-ear latency, which may easily exceed the strict delay requirements of NMP interactions.

This paper investigates the adoption of a low-complexity prediction technique based on autoregressive models to fill audio gaps caused by missing packets. Numerical results show that the proposed approach outperforms packet loss concealment methods normally implemented in NMP systems, typically based on filling audio gaps with silence or repetition of the last received audio segment.

CCS CONCEPTS

• **Networks** → **Application layer protocols**; **Error detection and error correction**; • **Applied computing** → **Sound and music computing**.

KEYWORDS

Networked music performances; packet loss concealment; autoregressive models

ACM Reference Format:
Matteo Sacchetto, Yuen Huang, Andrea Bianco, Cristina Rottondi. 2022. Using Autoregressive Models for Real-Time Packet Loss Concealment in Networked Music Performance Applications. In *AudioMostly 2022 (AM '22), September 6–9, 2022, St. Pölten, Austria.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3561212.3561226

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 musical interactions from remote locations by exploiting ultra-low latency audio/video streaming over the Internet. The Covid-19 pandemic is still spreading all around the world and social distancing and mobility restrictions between cities or countries are imposed to mitigate its diffusion. Thus, implementations of NMP applications have been gathering increasing attention in both academic and artistic communities.

To ensure a realistic musical interplay and good quality of experience in NMP, very strict requirements must be satisfied to keep the one-way end-to-end transmission latency below a few tens of milliseconds [7]. Using uncompressed audio streams and leveraging UDP (User Datagram Protocol) as transport layer protocol is a common solution to reduce time overhead in a real time application. Thus, the processing time of audio/video compression codecs and the delays introduced by the re-transmission mechanism implemented by TCP (Transmission Control Protocol) can be avoided at the price of sacrificing data transfer reliability. Indeed, the most critical limiting factor of distributed networked performances is the impact of the data loss and delay jitter introduced by IP networks. Due to the real time nature of NMP, an excessively delayed packet is treated as a lost packet, and it is dropped at the receiver. A lost audio packet causes a gap in the play-out audio which translates in an audio artifact or glitch. Since retransmission mechanisms or codec-based recovery methods are not applicable due to delay constraints, alternative recovery mechanisms must be devised to deal with packet losses to make the audio glitches not perceptible, by adequately filling the gaps introduced by missing portions of the play-out audio.

In this paper, we propose a Packet Loss Concealment (PLC) method implemented by using Autoregressive (AR) models. AR models are a special case of multi-linear regressive models, in which a forecast variable is obtained by using a linear combination of its past values. The idea is to predict and synthesize the lost data sections in the audio streams based on their historical trend, and to re-insert them in the stream to fill the gaps in the received audio stream before playback. Thanks to their low computational requirements, AR models represent a lightweight alternative to more complex prediction approaches (e.g., those based on Machine Learning techniques [10]) and can also be leveraged as benchmark to evaluate their performance.

We apply our proposed PLC technique to different musical genres and combinations of instruments, to evaluate their behaviour and to identify the parameter configuration that ensure the best trade-off between adequate prediction capabilities and computational requirements. To evaluate the accuracy of the model in generating the missing data, we selected two error metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). Furthermore, a qualitative analysis of the results was conducted through listening tests on a representative set of samples.

AM '22, September 6–9, 2022, St. Pölten, Austria

ACM ISBN 978-1-4503-9701-8/22/09…$15.00
https://doi.org/10.1145/3561212.3561226

Results show that AR models are able to closely predict the future audio samples for stationary data series, outperforming two classical PLC approaches widely employed in NMP applications: filling the audio gap with silence and replicating the audio samples contained in the last correctly received packet. In the literature [8], these two primitive methods are traditionally referred to as silence substitution and pattern replication. While these two approaches are widely used due to their negligible computational cost, in most cases they introduce significant distortions, resulting in audible glitches which, if relatively close to each other, may be perceived as noise.

The remainder of this paper is organized as follows: after briefly reviewing the related literature in Sec. 2, some background notions on AR models are provided in Sec. 3. The considered prediction framework and experimental setup are presented in Sec. 4 and numerical results are reported in Sec. 5. Conclusive remarks appear in the last Section.

2 RELATED WORK

The majority of the studies on PLC in NMP applications focus on recovery methods for the transmission of audio signals in MIDI format. The authors of [4] proposed a PLC method leveraging a recovery journal section within RTP packets, which holds the latest music status information, so that a receiver can recover from a wrong status caused by previous lost packets. Other recovery approaches rely on auxiliary reliably-connected channels to transmit critical MIDI events [11] or carry out the acknowledgment mechanisms to allow for re-transmission of lost music data [6].

For what concerns the prediction of the waveform of lost audio data, the authors of [12] adopt a state-of-the-art AR model for PLC in speech. The idea of predicting audio data by AR model in a speech can be applied in NMP too. In music-related contexts, AR models have already been applied for a variety of tasks. For example, they have been leveraged in combination with N-gram models to predict beat-synchronous Mel-frequency cepstral coefficients and chroma features in musical audio streams [3], and for online prediction of the tempo curve [5], where the parameters of the AR model are determined by a deep convolutional neural network fed with the past history of the tempo curve and the music score of the piece being performed. An AR self-attention based model for music score infilling is described in [2], with the aim of generating a polyphonic music sequence that fills in the gap between given past and future excerpts.

Differently, in [10], a deep convolutional neural network is adopted to conceal missing audio fragments. The proposed ML-based framework is shown to outperform the predictive capabilities of AR models, but would require GPU-accelerated hardware to ensure acceptable execution delays, which is at present unlikely to be available in NMP environments.

Moreover, there is a rich literature on ML-based generative models for raw audio, though not necessarily focused on NMP applications: the interested reader may refer to [1] for a comprehensive survey of the most relevant studies on such topic.

3 BACKGROUND

3.1 Autoregressive Models

The basic concept of time series regression models is that the models forecast the time series of a variable of interest y_t assuming that it has a linear relationship with another 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.

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

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_k x_{k,t} + \epsilon t. \quad (1)$$

The coefficients $\beta_0, \dots, \beta_k$ show the effect of each predictor $x_{k,t}$ on the forecast variable y_t after taking into account the effects of all the others, and must be estimated during the training phase of the model, together with an estimated minimum error ϵ.

AR models are a special case of multiple regression models. In an AR model, the forecast variable y_t can be expressed as a linear combination of its past values. In other words, AR models predict a variable of interest against the lagged values of itself. An autoregressive model of order ρ is denoted as AR(ρ) and can be written as:

$$y_t = c + \psi_1 y_{t-1} + \psi_2 y_{t-2} + \cdots + \psi_\rho y_{t-\rho} + \epsilon t. \quad (2)$$

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

3.2 Stationarity Test

An AR model is stable if and only if the considered time series is stationary. Indeed, since AR models predict future values on the basis of a linear combination of the past values, the model would diverge or have a biased trend for non-stationary time series. Since for audio traces both of these scenarios are not desirable, to use AR models for audio PLC, we first check if the considered time series is stationary. If the time series is not stationary, we must rely either on simple, traditional methods or on more complex alternative strategies, not covered in this paper.

To test the stationarity of the time series we adopt the Augmented Dickey-Fuller (ADF) test as our proof. The ADF test is a statistical significance test which tests the null hypothesis that a unit root is present in a time series. Under the assumption that the null hypothesis is correct, we evaluate the *p-value* (p), which can be defined as the probability of observing results which are as extreme or more extreme than the observed ones. We then compare it against the significance level α, which can be defined as the probability of rejecting the null hypothesis, and based on the obtained result we can conclude whether the null hypothesis is rejected, thus confirming the alternate hypothesis, or not. In particular, we consider the performed test as statistically significant when $p \leq \alpha$. The significance level α is traditionally set to 5% or lower, depending on the study. In our case α is set to 5%.

The *p-value* is a practical tool to evaluate the “strength of evidence” against the null hypothesis, but it can also be seen as the "strength of evidence" of confirming the alternative hypothesis, which in the case of the ADF test is the stationarity of the time series. Let us consider a simple AR(1) model defined as:

$$y_t = \psi_1 y_{t-1} + \epsilon t + c \tag{3}$$

where t is the time index, ϵ is white noise and c is a constant. We can write the regression model as:

$$\Delta y_t = (\psi_1 - 1) y_{t-1} + \epsilon t + c \tag{4}$$

where Δ is the first difference operator. We can conclude that the model has a unit root if $(\psi_1 - 1) = 0$, thus if $\psi_1 = 1$. The idea of the Dickey-Fuller test is to test if $\psi_1 = 1$ in Equation 3, which can be rewritten in a more general form as:

$$y_t = \psi_1 y_{t-1} + \Phi_1 \Delta y_{t-1} + \epsilon t + c \tag{5}$$

The ADF test is an augmented version of the Dickey-Fuller test, and it expands Equation 5 to include higher order regressions:

$$y_t = \psi_1 y_{t-1} + \Phi_1 \Delta y_{t-1} + \Phi_2 \Delta y_{t-2} + \cdots + \Phi_\rho \Delta y_{t-\rho} + \epsilon t + c \tag{6}$$

where ρ is the number of lags we are considering during the test. The unit root test is carried out under the null hypothesis $\psi_1 = 1$ against the alternate hypothesis $\psi_1 < 1$. If the *p-value* obtained through the ADF test, is < 0.05, then we can reject the null hypothesis concluding that the time series we are considering is stationary; if instead $p \geq 0.05$, the null hypothesis cannot be rejected and time series is considered non-stationary.

4 EXPERIMENTAL SETUP

4.1 Framework

To implement a PLC solution based on AR models, several steps must be executed. As already mentioned in Sec.3.2, the first step is the evaluation of the stationarity of time series. Once the stationarity evaluation is done, we proceed to tune the parameters of the AR model. We then evaluate its performance using the two error metrics: MAE and RMSE, and compare the results against the ones obtained using the two benchmark solutions used for PLC in a NMP context (silence substitution and pattern replication). This test is carried out with various audio files which belong to six different categories. Once we find the parameters which provide the best results across all the categories, we use them to generate short excerpts by artificially inserting audio gaps which are later used to perform a qualitative analysis via listening tests.

In the following subsections we will describe in more details each of the above-mentioned steps.

To better understand the content of the following subsections, we introduce some terms and symbols which will be used from now on:

- **training set** ($\mathcal{D}$): the section of the time series we are considering for fitting the AR model. It is the past history of the time series, that we will use to predict the future values. Its size will be identified by n (expressed in number of samples).
- **test set** ($\mathcal{D}_{test}$): the section of the time series that we want to compare our predicted values against. These are the actual future values of the time series, which will be used only to compute the prediction accuracy metrics. Its size will be identified by n_{test}.
- **position** (i): index in the array of samples of the time series, which represents the index of the first value we want to predict. The interval $[i - n, i)$ represents the section of the time series we are using as training set ($\mathcal{D}$), while the interval $[i, i + n_{test})$ represents the section of the time series used as test set ($\mathcal{D}_{test}$).

4.2 Stationarity Check

Since we are fitting our AR model on section $\mathcal{D}$ of the considered time series, the stationarity check does not need to be performed on the whole time series, but only on $\mathcal{D}$. So, depending on its size n we could obtain different results from the ADF test. Additionally, we know that the ADF test depends also on the number of lags (ρ) we are considering. In NMP, a typical size of a UDP packet payload, in terms of number of audio samples, is 128. Since we are trying to use AR model for PLC in NMP, we decided to only use multiples of 128 as the size of $\mathcal{D}$ (i.e., $n = k \cdot 128$ with k positive integer) and use exactly the size of one packet, i.e. 128, as the size of $\mathcal{D}_{test}$ (i.e., $n_{test} = 128$). This value allows us to generalize the proposed solution also to NMP web-applications that take advantage of the Web Audio API for processing, which operates on buffers of 128 samples. Regarding ρ, we start from 1 and double it up to 128, i.e., using as upper limit the number of samples in a single packet. n was set to 512, 1024, 2048, 4096 and 8192 samples, while ρ was set to all the powers of 2 between 1 and 128, extremes included.

To have a general understanding on the stationarity of each of the audio categories we identified, we carried out the stationarity test by looping through all the files of each category. For each audio file of a given category we then selected 100 random positions and for each position (i) we evaluated the stationarity of the time series for each combination of n and ρ. The implementation of this test is described with the pseudocode in Algorithm 4.1, where the ADF function is implemented using the `adfuller` function of the *statsmodels* Python module [9]. To quantify the stationarity of the model, we use the *p-value*. The lower the *p-value*, the more the time series is stationary, the higher the *p-value*, the less the time series is stationary. As for the ADF test specification, the series is considered stationary if the test result ≥ 0.05.

4.3 Training and testing of AR models

Also in this step we used the same criteria for selecting the values of n and ρ. However, as shown in the results in Sec. 5.2, for lower values of n, the time series tend to be non-stationary. Thus, for the training and testing of the AR models we omitted some values of n and considered only 2048, 4096 and 8192 samples, while ρ was set to all the powers of 2 between 1 and 128, as before. Again, n_{test} was set to 128.

In this phase, we aim to evaluate the performance of the AR models on each audio category. For this reason, we carried out this test by looping, for each audio category, through all the files belonging to it. For each audio file, we loaded all the samples; then, within all the samples we selected 100 positions and for each position (i), n and ρ, we trained the model and evaluated the two metrics (MAE and RMSE) by comparing the predictions with the actual data. For each i we also evaluated the two metrics for the two benchmark packet loss concealment methods used in NMP: silence

Algorithm 4.1: Stationarity check

```
$train\_sizes \leftarrow [512, 1024, 2048, 4096, 8192]$
$lags \leftarrow [1, 2, 4, 8, 16, 32, 64, 128]$

for $category$ in $categories$ do
    $p\_res \leftarrow []$

    for $file$ in $audio\_files$ do
        $samples \leftarrow load\_samples(file)$
        $positions \leftarrow generate\_100\_random\_indexes(samples)$

        for $tain\_size$ in $tain\_sizes$ do
            for $lag$ in $lags$ do
                for $pos$ in $positions$ do
                    $train\_set \leftarrow samples[pos - train\_size, pos]$
                    $p\_value \leftarrow ADF(train\_set, lag)$
                    $p\_res[train\_size][lag].append(p\_value)$
                end for
            end for
        end for
    end for

    $m \leftarrow mean(p\_res)$
    $interval \leftarrow confidence\_interval(p\_res)$
    $plot(m, interval)$
end for
```

substitution and pattern replication. The pseudocode relative to the implementation of this phase is shown in Algorithm 4.2, where the AR class is implemented using the `AutoReg` class of the *statsmodels* Python module [9].

4.4 Quality assessment

In this last phase, we identified the combination of n, ρ values that provided the best trade-off in terms of performance versus required computational resources, and generated a series of samples for a qualitative analysis. For each category we selected two audio files, for a total amount of 12 files. We manually trimmed all the audio files to be 5s long, by selecting the portion of the file which was most relevant from the musical point of view (e.g., excerpts with silence periods were avoided). For each audio file we then simulated the loss of 20 packets, by selecting 20 random positions. For each position (i) we removed the n_{test} samples in the interval $[i, i+n_{test})$. We then recovered the n_{test} samples by using the three PLC techniques: silence substitution, pattern replication and AR models. The three different audio files with recovered audio gaps were then stored on disk. Finally, we set up a quality test which involved human listeners: volunteers were recruited and asked to listen to the original excerpt and to the three generated audio files, and then to select the one which sounded more similar to the original. Algorithm 4.3 reports the pseudocode for the preparation of the excerpts.

Algorithm 4.2: Training and testing

```
$train\_sizes \leftarrow [2048, 4096, 8192]$
$lags \leftarrow [1, 2, 4, 8, 16, 32, 64, 128]$
$test\_size \leftarrow 128$

for $category$ in $categories$ do
    $mae\_res \leftarrow []$
    $rmse\_res \leftarrow []$

    for $file$ in $audio\_files$ do
        $samples \leftarrow load\_samples(file)$
        $positions \leftarrow generate\_100\_random\_indexes(samples)$

        for $pos$ in $positions$ do
            $silence \leftarrow [0] * test\_size$
            $previous \leftarrow samples[pos - test\_size, pos]$
            $test\_set \leftarrow samples[pos, pos + test\_size]$

            $mae\_res.append(MAE(test\_set, silence))$
            $rmse\_res.append(RMSE(test\_set, silence))$

            $mae\_res.append(MAE(test\_set, previous))$
            $rmse\_res.append(RMSE(test\_set, previous))$
        end for

        for $train\_size$ in $train\_sizes$ do
            for $lag$ in $lags$ do
                for $pos$ in $positions$ do
                    $train\_set \leftarrow samples[pos - train\_size, pos]$
                    $test\_set \leftarrow samples[pos, pos + test\_size]$
                    $start \leftarrow train\_size$
                    $end \leftarrow train\_size + test\_size - 1$

                    $AR\_model \leftarrow AR(train\_set, lag).fit()$
                    $pred \leftarrow AR\_model.predict(start, end)$

                    $mae \leftarrow MAE(test\_set, pred))$
                    $rmse \leftarrow RMSE(test\_set, pred))$

                    $mae\_res[train\_size][lag].append(mae)$
                    $rmse\_res[train\_size][lag].append(rmse)$
                end for
            end for
        end for
    end for

    $mae\_m \leftarrow mean(mae\_res)$
    $mae\_interval \leftarrow confidence\_interval(mae\_res)$
    $plot(mae\_m, mae\_interval)$

    $rmse\_m \leftarrow mean(mae\_res)$
    $rmse\_interval \leftarrow confidence\_interval(rmse\_res)$
    $plot(rmse\_m, rmse\_interval)$
end for
```

Algorithm 4.3: Quality assessment

```
$train\_size \leftarrow 2048$
$lag \leftarrow 64$
$test\_size \leftarrow 128$

for $category$ in $categories$ do
    for $file$ in $audio\_files$ do
        $samples \leftarrow load\_samples(file)$
        $b0\_samples \leftarrow samples.copy()$
        $b1\_samples \leftarrow samples.copy()$
        $ar\_samples \leftarrow samples.copy()$
        $positions \leftarrow generate\_20\_random\_indexes(samples)$

        for $pos$ in $positions$ do
            $train\_set \leftarrow samples[pos - train\_size, pos]$
            $test\_set \leftarrow samples[pos, pos + test\_size]$
            $start \leftarrow train\_size$
            $end \leftarrow train\_size + test\_size - 1$

            $silence \leftarrow [0] * test\_size$
            $prev \leftarrow samples[pos - test\_size, pos]$

            $b0\_samples[pos, pos + test\_size] = silence$
            $b1\_samples[pos, pos + test\_size] = prev$

            $AR\_model \leftarrow AR(train\_set, lag).fit()$
            $pred \leftarrow AR\_model.predict(start, end)$
            $ar\_samples[pos, pos + test\_size] = pred$
        end for

        $store\_samples(b0\_samples, b0\_file)$
        $store\_samples(b1\_samples, b1\_file)$
        $store\_samples(ar\_samples, ar\_file)$
    end for
end for
```

5 NUMERICAL ASSESSMENT

5.1 Dataset

Our dataset is a collection of 27 music files, consisting of 22 music files of five genres and five mixture music songs. All the audio files are sampled at the rate of 22050Hz and the number of samples of each file is at least 50000 and can reach 5000000. The audio files are grouped in six categories, four of which contain various samples of the relative individual instrument (Violin, Drums, Piano and Guitar), one (Generic) which contains samples of different individual instruments or vocals, and one which contains full songs.

5.2 Stationarity Test

As reported in Fig. 1, using $n = 512$ samples resulted in a non-stationary time series for most of the 6 categories. In general, the bigger n, the lower the average $p - value$. Thus, by increasing n the time series can be considered as stationary. Conversely, by increasing ρ, stationarity characteristics do not become so evident, and for each category a preferential value of ρ appears to minimize the $p - value$. It also emerges that 128 lags are excessive and cause the model to diverge, especially with low values of n. Since training sizes of 512 and 1024 samples provided unsatisfactory results, we decided to exclude them in the performance evaluation reported in the following subsections.

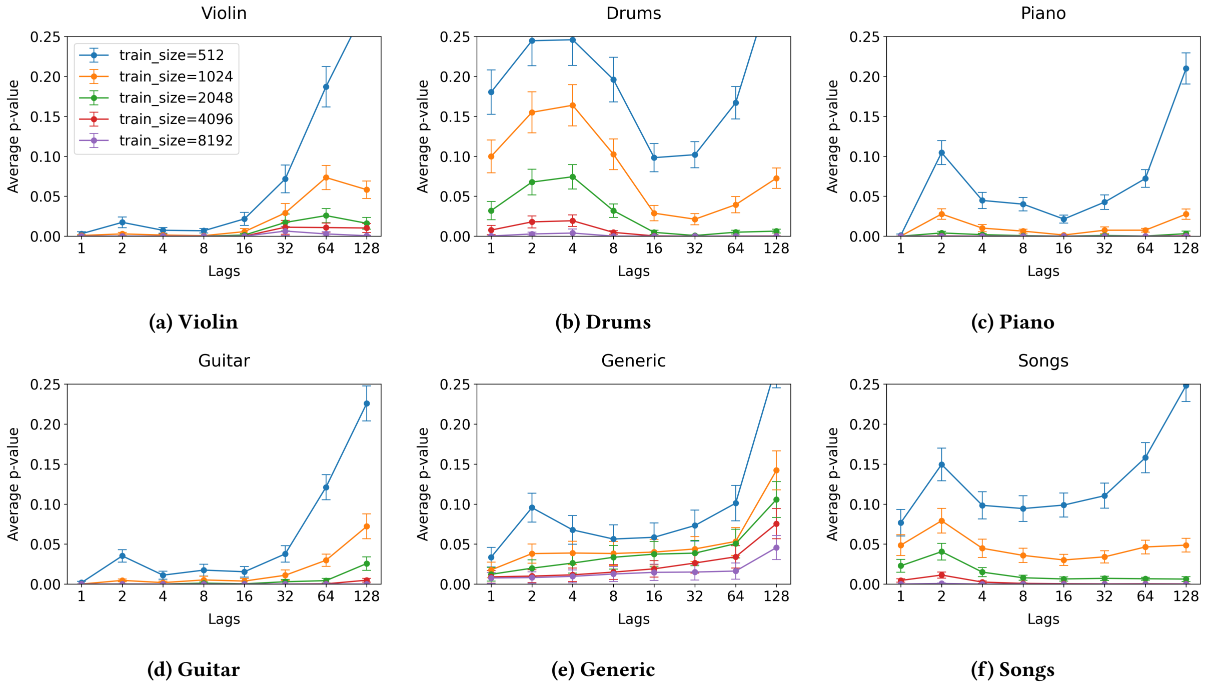
5.3 AR Model Results

In Fig. 2 and Fig. 3 the trend of MAE and RMSE is plotted for each category vs ρ, for variable n. The plots show also the performance of the two benchmarks PLC techniques: the yellow line represents the metric evaluation on the pattern replication solution, whereas the grey line represents the metric evaluation on the silence substitution solution. One apparently counterintuitive outcome is that, by replicating the previous packet, we obtain higher errors than those obtained with silence substitution. The reason is that, by replicating the previous packet, some discontinuity in the audio waveform is created. The replicated audio segment may be similar to the lost audio segment, but its phase is usually shifted. The shift creates a push-pull audio waveform which results in a higher difference between the actual samples and the replicated ones, thus resulting in a higher error with respect to that obtained with silence substitution. An example of the phase shift between the waveform generated by the pattern replication PLC technique and the actual waveform is visible in Fig. 4 where we show the resulting waveform created by the different PLC solution when reconstructing the same packet.

The plots show that ρ is the most crucial parameter to tune to improve the overall performance. On the other hand, n does not seem to significantly affect the performance of the AR models. In general, as long as our time series is stationary, the higher ρ and the lower n, the closer to zero the resulting error is. It follows that, if we want to deploy a PLC technique based on AR models, there is no need to store long sequences of historical audio data. Moreover, also the computational cost can be kept under control, since, even for lower number of lags, results show that the AR model already performs better than the two benchmarks.

5.4 Qualitative Analysis Results

The quantitative analysis reported in the previous subsections does not take into account that one of the most important requirements for a PLC technique in NMP is not to alter the perceived tempo. Indeed, when musicians play together, they derive an internal tempo reference from the perceived one and can continue playing following this internal tempo even if occasional silence periods are inserted in the audio playback they are listening to. If such silence periods are short enough, musicians are able to continue playing in sync, whereas longer silence periods may cause slight desynchronizations, that can be easily recovered once the audio play out is restored. On the other hand, replication of past audio samples may introduce alterations that, especially for percussive instruments, could be interpreted as double-beats, thus degrading the musicians' perception of the tempo curve, making their interplay more difficult. This is the main reason why the most widely adopted PLC solution in NMP is silence substitution. Being based on the past history of the time series, AR models have the property of not altering the musical tempo, while they should help to mitigate the impact of packet losses on the quality of the audio playback. To understand



(a) Violin

(b) Drums

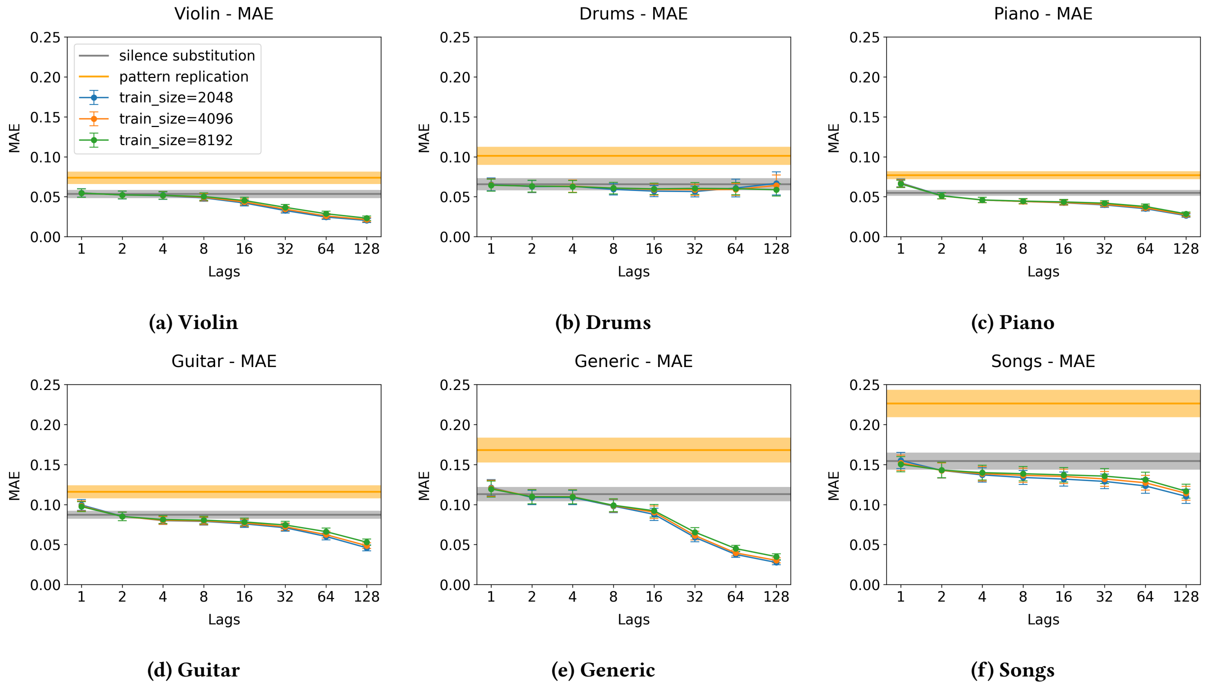
(c) Piano

(d) Guitar

(e) Generic

(f) Songs

Figure 1: Stationarity



(a) Violin

(b) Drums

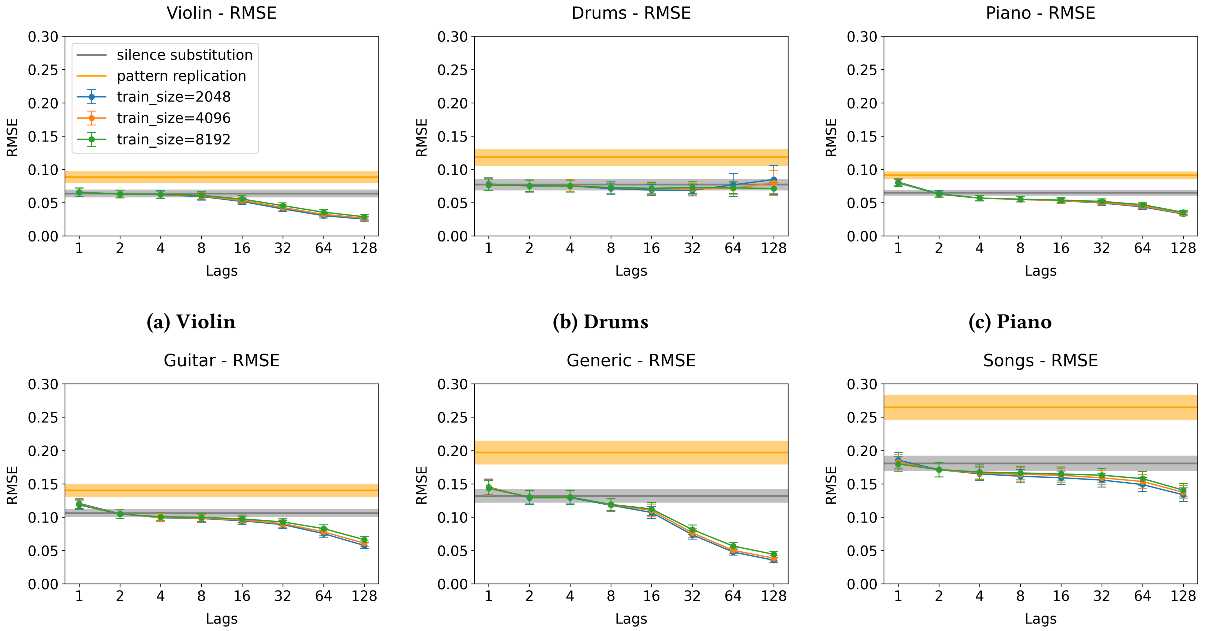
(c) Piano

(d) Guitar

(e) Generic

(f) Songs

Figure 2: AR models - MAE



(a) Violin

(b) Drums

(c) Piano

(d) Guitar

(e) Generic

(f) Songs

Figure 3: AR models - RMSE

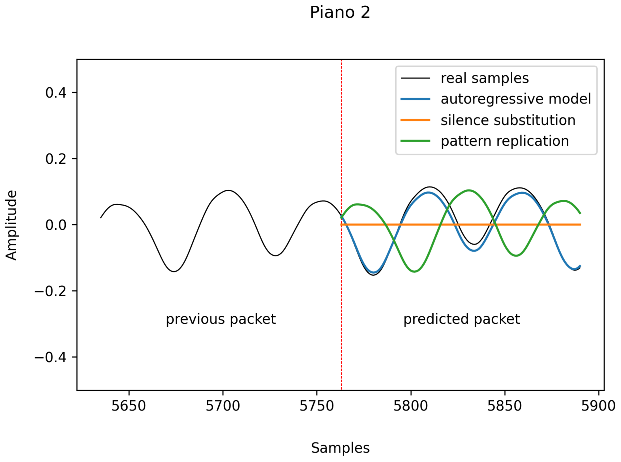


Figure 4: Visual comparison of the reconstructed waveform by the three PLC techniques

if such quality improvement was perceivable, we administered the test described in Sec. 4.4 to a group of 25 volunteers.

Results of this test can be found in Fig. 5, which shows that, in most cases, the recovered audio obtained with the PLC solution based on AR models was perceived to be the most similar to the original audio. The main difference people heard, from comments we received after the test, was in terms of noise. Testers selected the track which was perceived as "less noisy". The noise they refer to it is actually caused by signal discontinuities, which introduce audio glitches.

It is also worth noticing that, whenever we are replacing missing audio samples with predicted ones we are likely to create two discontinuities in the audio signal, at the two extremes of the audio gap. The perceived intensity of the glitches depends on the distance between the two consecutive samples that cause the discontinuities. Since AR models predict values based on the past history of the time series, they ensure a continuity on the left side of the predicted section, thus generating (on average) only half the number of glitches w.r.t. the other two solutions. This is the main reason why, in most of the cases, the PLC method based on AR was preferred.

From this analysis it also emerges that AR models work better with certain categories of instruments: in particular, the more the signal contains sustained sounds, the better the AR models work as a PLC technique, while the more the signal contains transient sounds with abrupt variations, the less noticeable is the benefit of AR models over the two benchmarks. This difference in behaviour also holds for the pattern replication PLC technique, which provides a better experience with respect to silence substitution for more sustained sounds, while it provides worse results for more transient-oriented sounds and may even create a flanging effect if the lost audio section extends for more than one audio packet.

The results of the test showed that for all the categories that include single instrumental sources, the AR models solution was preferred, while for the category ’Songs’ that includes polyphonic music, no clear preference emerges. Indeed, polyphonic music masks better audio glitches, thus making the differences between the three solutions less noticeable.

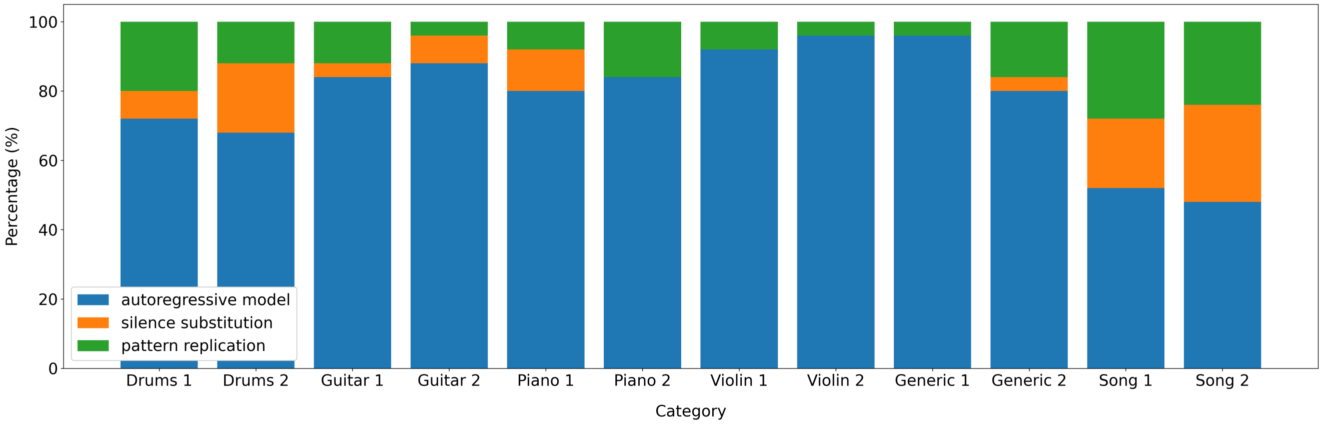


Figure 5: Qualitative analysis results

6 CONCLUSION

In this paper we propose a PLC solution based on AR models and targeted to NMP applications. Experiments were performed considering various categories of audio signals and involved a preliminary analysis on the stationarity of the time series, followed by a quantitative analysis on the performance of the AR models w.r.t. to two benchmark PLC solutions (silence substitution and pattern replication) and lastly by a qualitative analysis of the quality of the reconstructed audio.

Results showed that, if the time series used to fit the AR models is stationary, the proposed method outperforms the two benchmarks in terms of mean absolute error and root mean square error. These results were also confirmed by the qualitative analysis which showed that, especially for solo instruments, AR models are perceived to recreate an audio track more similar to the original one in comparison to the two benchmarks. Additional improvements may be addressed in future works, such as the dynamic selection of the best number of lags directly from the reproduced audio material, or the adoption of signal processing techniques to impose a signal continuity to both ends of the generated section. Future work may also involve implementing the proposed solution in a NMP software, to measure its computational requirements in a real scenario.

REFERENCES

[1] Jean-Pierre Briot. 2021. From artificial neural networks to deep learning for music generation: history, concepts and trends. *Neural Computing and Applications* 33, 1 (2021), 39–65.

[2] Chin-Jui Chang, Chun-Yi Lee, and Yi-Hsuan Yang. 2021. Variable-length music score infilling via XLNet and musically specialized positional encoding. *arXiv preprint arXiv:2108.05064* (2021).

[3] Peter Foster, Anssi Klapuri, and Mark D Plumbley. 2011. Causal Prediction of Continuous-Valued Music Features.. In *ISMIR*. Citeseer, 501–506.

[4] John Lazzaro and John Wawrzynek. 2001. A case for network musical performance. In *Proc. of the 11th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*. 157–166.

[5] Akira Maezawa. 2019. Deep linear autoregressive model for interpretable prediction of expressive tempo. *Proc. SMC* (2019), 364–371.

[6] Jason Robert Nelson, Allen James Heidorn, and Robert John Cox. 2017. Reliable real-time transmission of musical sound control data over wireless networks. US Patent 9,601,097.

[7] Cristina Rottondi, Chris Chafe, Claudio Allocchio, and Augusto Sarti. 2016. An Overview on Networked Music Performance Technologies. *IEEE Access* 4 (2016), 8823–8843.

[8] Henning Sanneck, Alexander Stenger, K Ben Younes, and Bernd Girod. 1996. A new technique for audio packet loss concealment. In *Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference*. IEEE, 48–52.

[9] Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and statistical modeling with Python. In *Proc. of the 9th Python in Science Conference*.

[10] Prateek Verma, Alessandro Ilic Mezza, Chris Chafe, and Cristina Rottondi. 2020. 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)*. 268–275. https://doi.org/10.23919/FRUCT49677.2020.9210988

[11] Jussi Virolainen and Pauli Laine. 2005. Methods and apparatus for transmitting MIDI data over a lossy communications channel. US Patent 6,898,729.

[12] Guoqiang Zhang and W Bastiaan Kleijn. 2008. Autoregressive model-based speech packet-loss concealment. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 4797–4800.