Vol.2017-SLP-115 No.2
2017/2/17

Investigation on an autoregressive recurrent mixture density network for parametric speech synthesis

Xin Wang[1,a)] Shinji Takaki[1,b)] Junichi Yamagishi[1,c)]

Abstract: Neural-network-based mixture density networks are important tools for acoustic modeling in statistical parametric speech synthesis. Recently we found that incorporating an autoregressive model in a recurrent mixture density network, which is referred to as AR-RMDN, enabled the network to generate quite smooth acoustic data trajectories without using the delta and delta-delta coefficients. More interestingly, the new model generated trajectories with a dynamic range similar to that of the natural data, thus alleviating over-smoothing effect. In this work, after explaining the AR-RMDN from the perspective of signal and filter, we compare one AR-RMDN with a modulation-spectrum-based post-filtering method that also eases the over-smoothing effect. It is demonstrated that the AR-RMDN also alters the modulation spectrum of the generated data trajectories but in a different way from the post-filtering method. The AR-RMDN also generates synthetic speech with better perceived quality. Based on the signal and filter interpretation, we further extend the AR-RMDN so that the inverse AR filter can acquire complex poles and stay stable.

Keywords: Text-to-Speech Synthesis, Recurrent Neural Network, Mixture Density Network, Autoregressive Model

1. Introduction

Statistical parametric speech synthesis (SPSS) uses statistical models to convert the input textual features into the acoustic features, based on which a speech waveform can be constructed using a vocoder. A classical SPSS framework is based on the hidden Markov model (HMM) [1]. Recently, it was augmented with various neural-networks-based methods [2]. For example, neural networks can be directly used to transform the input textual feature into the target acoustic feature frame by frame [3], [4], [5], [6]. Alternatively, neural networks can be formulated as a generative model such as the mixture density network (MDN) [7], [8]. This type of neural network transforms the input features into a parameter set of the acoustic feature's distribution at each frame. The acoustic feature trajectories then can be generated from the sequence of distributions in a similar way to the process after the sequence of HMM state is determined in the HMM-based SPSS [9].

In either case, an ideal neural-network-based model should generate the acoustic feature trajectories that are sufficiently smooth yet not over-smoothed. Unfortunately, the conventional neural networks cannot easily reproduce the natural evolution of acoustic feature trajectories. A typical difficulty is on the temporal correlation across adjacent frames [10], even for some recurrent neural networks [11]. To generate smooth trajectories, the delta & delta-delta coefficients of the acoustic feature can be modeled and used explicitly [3] or implicitly [12]. Another intrinsic shortcoming of most statistical synthesizers is the over-smoothing effect, i.e., the model generates data that are close to the mean of the distribution. Effective methods to alleviate the over-smoothing problem include the training or generation process with the global variance (GV) [12], [13] and the modulation spectrum (MS) into consideration [14], [15].

Alternatively, we recently found that the recurrent mixture density network with an autoregressive model, which is referred to as Autoregressive Recurrent Mixture Density Model (AR-RMDN), can alleviate the over-smoothing effect without using GV or MS explicitly. Furthermore, the generated trajectories can be smooth without using delta & delta-delta coefficients [8]. In this work, we explain the way that the AR-RMDN generate the acoustic feature trajectory with its MS enhanced. We also compare one AR-RMDN with a MS-based poster-filtering method, and show the better performance of the AR-RMDN. Besides, we extend the AR-RMDN so that the AR filter in the synthesis stage can acquire complex poles while be stable.

In the rest of paper, Section 2 defines and explains the AR-RMDN; Section 3 shows the way to formulate a stable complex inverse AR filter; Section 4 includes the experiment comparing AR-RMDN and MS-based post-filtering method, and experiment on using the extended AR-RMDN for F0 modeling.

2. Define and Interpret the AR-RMDN

2.1 Review of Recurrent Mixture Density Networks

The basic component of AR-RMDN is the recurrent mixture density network (RMDN). Similar to the normal recurrent neural network (RNN), an RMDN uses a recurrent layer to compute the hidden feature $\boldsymbol{h}_t$ at the time t given the input $\boldsymbol{x}_t$ to this layer and the previously extracted hidden feature $\boldsymbol{h}_{t-1}$:

$$\boldsymbol{h}_t = \mathcal{F}(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{W}_h\boldsymbol{h}_{t-1} + \boldsymbol{d}). \quad (1)$$

1 National Institute of Informatics, Chiyoda 2-1-2, Tokyo 101–8430, Japan
a) wangxin@nii.ac.jp
b) takaki@nii.ac.jp
c) jyamagis@nii.ac.jp

Here, $\mathcal{F}$ is the activation function such as the *sigmoid* or *tanh*; $\boldsymbol{W}_*$ denotes the transformation matrix and $\boldsymbol{d}$ is the bias. The extracted $\boldsymbol{h}_t$ can be delivered to the next layer as input until the output layer where it is transformed into the final output. Note that Equation (1) can be replaced by the computation flow in a long-short term memory (LSTM) unit [16].

However, while an RNN directly uses the final output as the estimated target data $\{\widehat{\boldsymbol{o}_1}, \cdots, \widehat{\boldsymbol{o}_T}\}$, an RMDN assumes the output as the value of a parameter set based on which the probability density function (PDF) of the target data can be fully specified. For example, an RMDN may use the Gaussian mixture model (GMM) as the PDF of the target data $\boldsymbol{o}_t$

$$p(\boldsymbol{o}_t; \mathcal{M}_t) = \sum_{m=1}^{M} \omega_t^m \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_t^m, \boldsymbol{\Sigma}_t^m). \tag{2}$$

Here, M is the number of mixture components and $\mathcal{M}_t = \{\omega_t^1, \cdots, \omega_t^M, \boldsymbol{\mu}_t^1, \cdots, \boldsymbol{\mu}_t^M, \boldsymbol{\Sigma}_t^1, \cdots, \boldsymbol{\Sigma}_t^M\}$ is the parameter set whose value is given by the output of the RMDN [17].

For the RMDN in a parametric speech synthesis system, the target sequence $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$ denotes the sequence of acoustic feature vectors while the input is the sequence of textual feature vectors. A GMM with diagonal covariance matrix $\boldsymbol{\Sigma}_t^m$ is commonly used. The RNN can be also formulated as an RMDN that uses a Gaussian distribution with a unity covariance matrix.

2.2 Definition of the AR-RMDN

Based on Equation (2), the likelihood of an acoustic feature sequence $\boldsymbol{o}_{1:T}$ on an RMDN can be written as

$$p(\boldsymbol{o}_{1:T}; \mathcal{M}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t | \mathcal{M}_t). \tag{3}$$

This equation assumes that $\boldsymbol{o}_t$s are statistically independent, and it ignores the temporal evolution of $\boldsymbol{o}_{1:T}$. Even for the RMDN where $\mathcal{M}_t, t \in \{1, \cdots, T\}$ are somehow related by the recurrent computation, statistical correlations are ignored among $\boldsymbol{o}_{1:T}$. One remedy is to use the delta coefficients as additional target data, and then use the 'maximum likelihood parameter generation' (MLPG) method to generate smooth trajectories [9]. Unfortunately, this method yields an invalid statistical model [18]. A solution is to use the trajectory model [12], which not only forms normalized model but also reproduces smooth acoustic feature trajectories.

Different from the above approaches, the AR-RMDN assumes that acoustic data vectors in the previous K time steps $\boldsymbol{o}_{t-K:t-1} = \{\boldsymbol{o}_{t-K}, \cdots, \boldsymbol{o}_{t-1}\}$ affect the component means of the GMM at the current time t. It accordingly defines the PDF of $\boldsymbol{o}_t$ as

$$p(\boldsymbol{o}_t | \boldsymbol{o}_{t-K:t-1}; \mathcal{M}_t) = \sum_{m=1}^{M} \omega_t^m \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_t^m + f(\boldsymbol{o}_{t-K:t-1}), \boldsymbol{\Sigma}_t^m), \tag{4}$$

where $f(\boldsymbol{o}_{t-K:t-1}) : \mathbb{R}^{K\times D} \to \mathbb{R}^D$ transforms $\boldsymbol{o}_{t-K:t-1}$ as

$$f(\boldsymbol{o}_{t-K:t-1}) = \sum_{k=1}^{K} \boldsymbol{a}_k \odot \boldsymbol{o}_{t-k} + \boldsymbol{b}. \tag{5}$$

Here, $\boldsymbol{b} \in \mathbb{R}^D$ is the bias, $\boldsymbol{a}_k \in \mathbb{R}^D$ is a vector that scales $\boldsymbol{o}_{t-k} \in \mathbb{R}^D$ by element-wise product $\odot$ and D is the dimension of the vector. Note that $\boldsymbol{o}_t = \boldsymbol{0}, t \leq 0$.

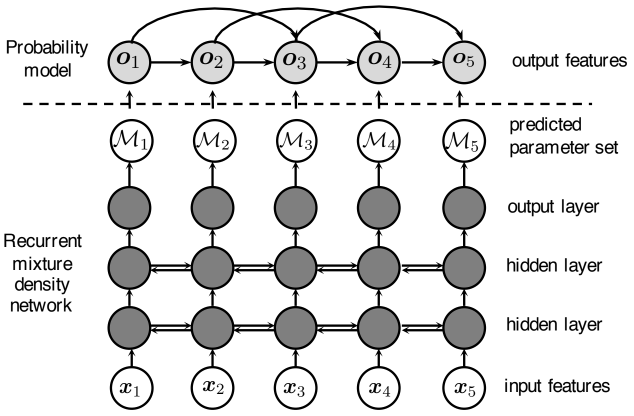

Fig. 1: An autoregressive recurrent mixture density network

In our implementation, $\boldsymbol{a}_k$ and $\boldsymbol{b}$ are time-invariant and shared among mixture components of the GMM. In the training stage, $\boldsymbol{a}_k, \boldsymbol{b}$ and the neural network weight can be tuned using the back-propagation algorithm under the maximum likelihood criterion. In the generation stage, the acoustic feature vectors can be generated sequentially. First, the distribution $p(\boldsymbol{o}_t | \boldsymbol{o}_{t-K:t-1}; \mathcal{M}_t)$ can be determined given the previously generated vectors $\hat{\boldsymbol{o}}_{t-K:t-1}$. Then the vector for time t can be generated as $\hat{\boldsymbol{o}}_t = \boldsymbol{\mu}_t^{m^*} + f(\boldsymbol{o}_{t-K:t-1})$ where $m^* = \arg\max_m w_t^m$.

2.3 Interpretation on the AR-RMDN

An AR-RMDN with K=2 is illustrated in **Fig. 1**. Similar to the Autoregressive HMM [19], AR-RMDN only assumes the distribution is affected by the past observations. This assumption is different from the methods using delta coefficients or trajectory models [12], [18]. This uni-directional dependence makes the model both efficient in computation and consistent in the statistical sense [19].

Another interesting angle to interpret the AR-RMDN is based on the signal and filter theory. As the covariance matrix of the GMM is diagonal, we can re-write Equation (4) as

$$p(\boldsymbol{o}_t | \boldsymbol{o}_{t-1}; \mathcal{M}_t) = \sum_{m=1}^{M} w_t^m \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi {\sigma_{t,d}^m}^2}} \exp(-\frac{o_{t,d} - \sum_{k=1}^{K} a_{k,d} o_{t-1,d} - \mu_{t,d}^m - b_d}{2{\sigma_{t,d}^m}^2}), \tag{6}$$

where $o_{t,d}, \mu_{t,d}^m, a_{k,d}$, and b_d are the d-th dimension of $\boldsymbol{o}_t, \boldsymbol{\mu}_t^m$, $\boldsymbol{a}_k$, $\boldsymbol{b}$, respectively, and $\sigma_{t,d}^m$ is the d-th element of the diagonal $\boldsymbol{\Sigma}_t^m$. If we define a new variable $c_{t,d} = o_{t,d} - \sum_{k=1}^{K} a_{k,d} o_{t-1,d}$, we can find that the sequence $\boldsymbol{o}_{1:T,d} = [o_{1,d}, \cdots, o_{T,d}]^\top$ and $\boldsymbol{c}_{1:T,d} = [c_{1,d}, \cdots, c_{T,d}]^\top$ is related by a linear transformation:

$$\boldsymbol{c}_{1:T,d} = \boldsymbol{A}^{(d)} \boldsymbol{o}_{1:T,d} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -a_{1,d} & 1 & 0 & 0 & \cdots & 0 & 0 \\ -a_{2,d} & -a_{1,d} & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & -a_{K,d} & \cdots & -a_{1,d} & 1 \end{bmatrix} \begin{bmatrix} o_{1,d} \\ o_{2,d} \\ o_{3,d} \\ \vdots \\ o_{T,d} \end{bmatrix}. \tag{7}$$

From the perspective of signal and filter, the linear transformation above is also a filtering operation. It is easy to show that the filter specified by the matrix $\boldsymbol{A}^{(d)}$ can be written in the z-domain as

$$A^d(z) = 1 - \sum_{k=1}^{K} a_{k,d} z^{-k}. \tag{8}$$

The input 'signal' $\boldsymbol{o}_{1:T,d}$ is filtered by this finite impulse response (FIR) filter $A^d(z)$. In this work, we refer to $A^d(z)$ as the *AR filter*.

Based on the interpretation above, we can derive the filtered 'signal' $\boldsymbol{c}_{1:T,d}$ for $d \in [1, D]$ and then re-write the Equation (4) with respect to random variable $\boldsymbol{c}_t$:

$$p(\boldsymbol{o}_{1:T}; \mathcal{M}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t | \boldsymbol{o}_{t-K:t-1}; \mathcal{M}_t) = \prod_{t=1}^{T} p(\boldsymbol{c}_t; \mathcal{M}_t). \tag{9}$$

This formulation shows that the parameter set $\mathcal{M}_{1:T}$ stays the same for $\boldsymbol{c}_t$. Besides, it indicates that:

- In the training stage, AR-RMDN uses the AR filter $A^d(z), d \in [1, D]$ to filter the original acoustic data $\boldsymbol{o}_{1:T,d}$ in each dimension $d \in [1, D]$. Then, it uses the RMDN part to describe the distribution of the filtered data $\boldsymbol{c}_{1:T,d}, d \in [1, D]$;
- In the synthesis stage, AR-RMDN uses the RMDN part to generate the acoustic vector $\hat{\boldsymbol{c}}_{1:T}$, and then uses the inverse AR filter $H^d(z) = 1/A^d(z)$ to de-transform $\hat{\boldsymbol{c}}_{1:T,d}$ into $\hat{\boldsymbol{o}}_{1:T,d}, d \in [1, D]$.

As we found in previous work, the AR filter processes the training data so that it can be better modeled by the RMDN part, while the inverse AR filter enhances the trajectories generated from RMDN sub-network.

3. Extend the AR Filter with Complex Poles

The inverse AR filter $H(z) = 1/A(z)$ in the synthesis stage should be stable [*1]. For $H(z)$ in AR-RMDN, our previous method [8] re-writes the $H(z)$ as a cascade

$$H(z) = \frac{1}{1 - \sum_{k=1}^{K} a_k z^{-k}} \tag{10}$$

$$= \prod_{k=1}^{K} \frac{1}{1 - \alpha_k z^{-1}}, \tag{11}$$

and then constrains α_k to be the output of a *tanh* function as

$$\alpha_k = \tanh(\hat{\alpha}_k). \tag{12}$$

The use of the *tanh* function ensures that the poles of $H(z)$ are located within [-1,1] on the real axis, which makes the filter stable.

However, the above assumption ignores the fact that a filter could acquire one or more pairs of conjugate complex poles. Although a complex-valued neural network could be a potential solution to handle complex numbers, here we stick to the real-valued network and propose a method to constrain the location of complex poles. This method assumes that a K-order filter $H(z)$ can be written as

$$H(z) = \begin{cases} \frac{1}{\prod_{k=1}^{K/2}(1-\alpha_k z^{-1} - \beta_k z^{-2})} & \text{if } K \text{ is even} \\ \frac{1}{(1-\alpha_0 z^{-1})\prod_{k=1}^{(K-1)/2}(1-\alpha_k z^{-1} - \beta_k z^{-2})} & \text{if } K \text{ is odd.} \end{cases} \tag{13}$$

Similar to the previous method, the pole α_0 can be constrained

[*1] To simplify the notation, the dimension index d is dropped

by a *tanh* function. For the other sub-filters, we assume that their poles $[\frac{\alpha_k + \sqrt{\alpha_k^2 + 4\beta_k}}{2}, \frac{\alpha_k - \sqrt{\alpha_k^2 + 4\beta_k}}{2}]$ must be a pair of conjugated complex poles inside the unit circle, which means that the parameters α_k and β_k must satisfy the conditions

$$\alpha_k^2 + 4\beta_k \in (-\infty, 0), \tag{14}$$

$$\left\| \frac{\alpha_k \pm \sqrt{\alpha_k^2 + 4\beta_k}}{2} \right\|_2^2 \in (0, 1). \tag{15}$$

The first condition constrains the poles to be complex numbers while the second condition constrains the location of the poles. Based on the first condition, the left-hand side of the second condition can be written as

$$\left\| \frac{\alpha_k}{2} \pm \frac{\sqrt{-(\alpha_k^2 + 4\beta_k)}}{2} i \right\|_2^2 = \frac{\alpha_k^2}{4} - \frac{\alpha_k^2 + 4\beta_k}{4} = -\beta_k, \tag{16}$$

where $i = \sqrt{-1}$. Then the conditions can be re-written as

$$\alpha_k \in (-2\sqrt{-\beta_k}, 2\sqrt{-\beta_k}), \tag{17}$$

$$\beta_k \in (-1, 0). \tag{18}$$

In the neural network, the above condition can be met by setting

$$\beta_k = -\text{sigmoid}(\widehat{\beta_k}), \tag{19}$$

$$\alpha_k = 2\sqrt{\text{sigmoid}(\widehat{\beta_k})}\tanh(\widehat{\alpha_k}). \tag{20}$$

Now, instead of the a_k in Equation (10), $\widehat{\beta_k}$ and $\widehat{\alpha_k}$ becomes the parameter of the AR filter that will be learned from the training data. In the back-propagation stage, the gradient with respect to $\widehat{\beta_k}$ and $\widehat{\alpha_k}$ can be computed using the chain rule in a straightforward way.

4. Experiments

4.1 Corpus and Systems

The corpus and configuration were the same as our previous work [8]. The corpus is the Blizzard Challenge 2011 corpus that has 12072 English utterances [20]. Both the test and validation set contained 500 randomly selected utterances. Mel-generalized cepstral coefficients (MGCs) of order 60, continuous F0 trajectory, voiced/unvoiced (V/U) condition, and band aperiodicities (BAP) of order 25 were extracted for each speech frame by using the STRAIGHT vocoder [21]. The Flite toolkit [22] conducted the text analysis for the entire corpus. The output of Flite was converted into a vector of dimension 382 as the input to the neural network.

Two systems were investigated: `RMDN` and `AR-RMDN`. Neither system used the delta-coefficients and the MLPG generation algorithm. Both `RMDN` and `AR-RMDN` included 2 feedforward layers with 512 nodes, 2 bi-directional LSTM layers with 256 nodes, and an output MDN layer. This MDN layer contained a binomial distribution for the V/U condition, three GMMs for the MGC, F0, and BAP with the number of mixture components as 2, 2, 1, respectively.

The training recipe include 3 steps. First, a conventional RNN network was trained; then, `RMDN` was initialized using the trained RNN and fine-tuned on the corpus; finally, the `AR-RMDN` was fine-tuned after being initialized by the trained `RMDN`.

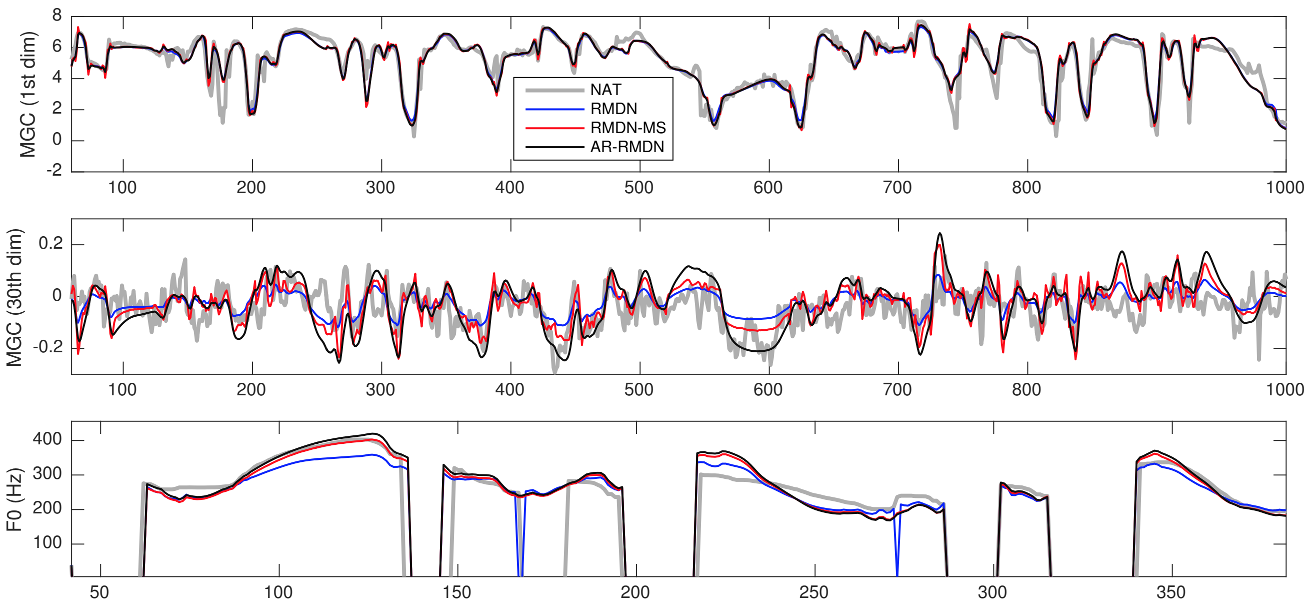

Fig. 2: Predicted MGC (1st and 30th order) and F0 trajectories against the natural data (NAT). The x-axis is the frame index. The plotted F0 was obtained using the predicted continuous F0 and U/V condition.

4.2 Comparing AR-RMDN with MS-based post-filtering

In this experiment, we focuses on the comparison between AR-RMDN and a related post-filtering approach, and only consider the AR filter with one real pole. The `AR-RMDN` network set the order of the AR filter as $K = 1$ for both MGC and F0 streams. The `AR-RMDN` is formulated without the constraint on the inverse AR filter according to Equation (10). The parameter $\{\boldsymbol{a}_k, \boldsymbol{b}\}$ was initialized as zero. In the generation stage, `RMDN` and `AR-RMDN` used the mean of the most probable mixture component as the output for each frame. [*2]

We plotted the trajectories for the 1st and 15th dimension of the MGC and F0 in **Fig. 2**. Comparing only the `RMDN` and `AR-RMDN`, we can see that, while the trajectories of both `AR-RMDN` and `RMDN` follows the natural data, the `AR-RMDN` one has a dynamic range that is closer to the natural (NAT) variance. If the global variance (GV) is used to measure the dynamic range of the generated features, the generated data from `AR-RMDN` acquires a higher GV. As researchers argued that the sum of linear-scaled modulation spectrum (MS) except bias is equivalent to GV [14], we analyzed the MS of the generated trajectories from `AR-RMDN` and `RMDN`. The results are plotted in **Fig. 3**. This feature clearly shows that the trajectory given by `AR-RMDN` has higher energy than `RMDN` in the low-frequency region. This is consistent with what the trajectories show in Fig. 2, namely the trajectory of `AR-RMDN` is dynamic yet smooth.

What contributes to the performance of `AR-RMDN` is the inverse AR filter $H^d(z), d \in [1, 60]$. **Fig. 4** plots the frequency response of $H^d(z)$ for certain dimensions of the MGC and F0. Clearly, the trained $H^d(z)$ enhances the low-frequency variation of the generated trajectories given by the RMDN sub-network of `AR-RMDN`. Due to this enhancement, the energy of the low-frequency region in the modulation spectrum of the `AR-RMDN`'s

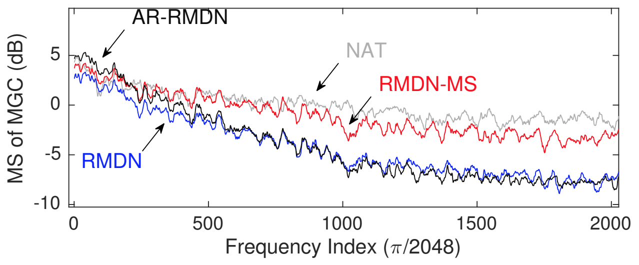

Fig. 3: Modulation spectrum of the 30th dim MGC.

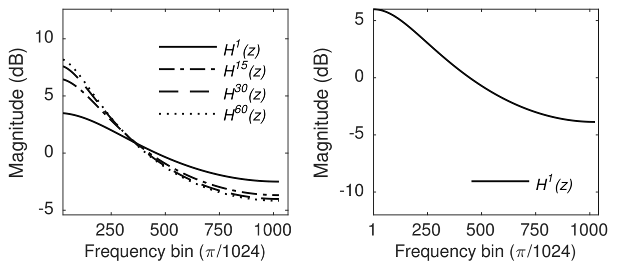

Fig. 4: Frequency responses of $H^d(z)$ in the trained AR-RMDN for the MGC (left) and F0 (right).

output can be larger than `RMDN`.

The results above show that `AR-RMDN` suffers less from the over-smoothing problem than `RMDN`. To deal with the over-smoothing problem, the modulation-spectrum (MS)-based post-filtering method has been proposed and achieved good results. Because MS-based post-filtering method directly manipulates the MS of the generated feature trajectories in the whole modulation frequency domain, it is different from what `AR-RMDN` does. Thus, we compared the two. For the MS-based systems, we trained the post-filter on the training data and then used it to enhance the output from the `RMDN`. Experiment groups were constructed according to **Table 1**. A subjective evaluation based on MUSHRA method with 10 paid native English listeners was conducted at the University of Edinburgh.

[*2] The toolkit is a modified version of CURRENNT [23]. This toolkit, implementation details of `AR-RMDN`, and speech samples can be found at http://tonywangx.github.io

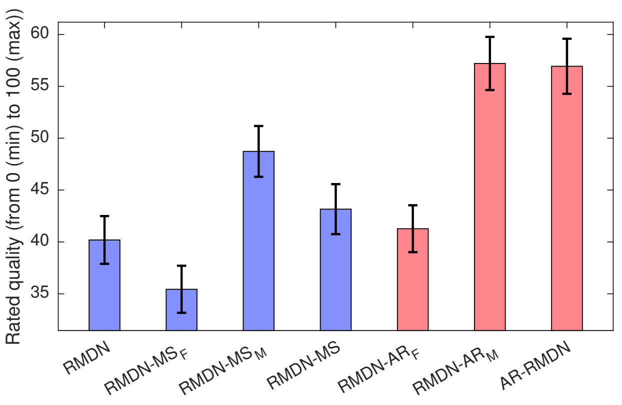

Fig. 5: Results of a MUSHRA test comparing AR-RMDN and MS-based post-filtering method (notation defined in Table 1).

Table 1: Experimental groups to compare AR-RMDN with MS-based post-filtering method in Fig.5

Notation	Description
`RMDN`	baseline samples from `RMDN`
`RMDN-MS`$_F$	`RMDN` with F0 enhanced by MS-based post-filter
`RMDN-MS`$_M$	`RMDN` with MGC enhanced by MS-based post-filter
`RMDN-MS`	`RMDN` with F0 and MGC enhanced by MS-based post-filter
`RMDN-AR`$_F$	baseline samples yet with F0 generated by `AR-RMDN`
`RMDN-AR`$_M$	baseline samples yet with MGC generated by `AR-RMDN`
`AR-RMDN`	samples from `AR-RMDN`

The results are shown in **Fig. 5**. The subjective evaluation shows that the `AR-RMDN` achieved better quality than the MS-based method. A particular artifact in the MS-enhanced speech is 'click' sounds. This type of click sound may come from the high-frequency noise introduced by the MS-based post-filtering. This can be seen by the trajectory of `RMDN-MS` plotted in Fig. 2. Another possible reason is that the 'phase' of the original trajectory was used to reconstruct the enhanced trajectory in the MS-based method. `AR-RMDN` may avoid these two problems as it only focuses on the low-frequency region and enhances the trajectory in the 'time domain' without transforming the trajectories in and from the modulation spectrum domain.

4.3 High-order inverse AR filter on F0 modeling

In the previous section, we showed the performance of the `AR-RMDN` with $K = 1$ for both MGC and F0. In this section, we explore its performance when $K > 1$. Because the dimension of MGC is much higher than F0, we only focus on F0 modeling for this initial trial. The `AR-RMDN` networks adopted the same configuration in previous section except the order and form of the AR filter. The networks are listed in **Table 2**.

Figure 6 shows the pole locations for networks $\mathtt{C}_6$, $\mathtt{R}_6$ and $\mathtt{U}_6$. As expected, while $\mathtt{R}_6$'s poles located between (-1,1) on the real-axis, $\mathtt{C}_6$ acquired complex poles within the unit circle. On the other hand, $\mathtt{U}_6$ had one pole outside the unit-circle, which means the inverse AR filter $H(z)$ is unstable. This result shows that the proposed method is useful to constrain the pole locations.

The generated F0 trajectories are shown in **Fig. 7**. An informal subjective evaluation showed that higher order AR doesn't improve the F0 over the $\mathtt{U}_1$. On one hand, the high order

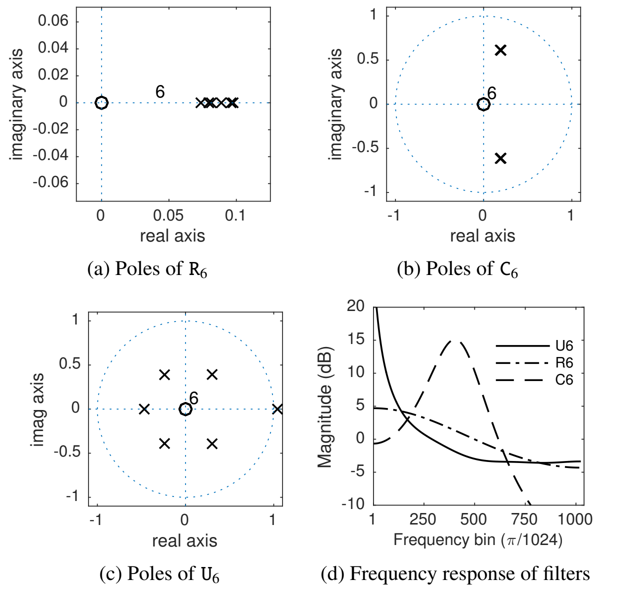

Fig. 6: Pole locations (marked by ×) and frequency responses of $H^d(z)$ for F0. Note that the three pairs of poles in (b) locates very close to each other.

Table 2: Notation of `AR-RMDN` in Section 4.3. All the networks used the same configuration as in Section 4.2 except the order and form of the AR filter for F0 (Eq. means equation).

Order of AR filter					
1	2	4	6	Form	Constraints
-	$\mathtt{C}_2$	$\mathtt{C}_4$	$\mathtt{C}_6$	Eq.(13)	Complex poles, stable (Eqs. (19)(20))
-	$\mathtt{R}_2$	$\mathtt{R}_4$	$\mathtt{R}_6$	Eq.(11)	Real poles, stable (Eq. (12))
$\mathtt{U}_1$	-	-	$\mathtt{U}_6$	Eq.(10)	Unconstrained

inverse AR with real poles ($\mathtt{R}_6$) is just a cascade of multiple 1st order filters, which performed similarly to the 1st order's case ($\mathtt{U}_1$). On the other hand, the AR with complex roles ($\mathtt{C}_6$) generated F0 with unnatural ripples, which can be clearly seen from Fig. 7. This can be explained that, as $\mathtt{R}_6$ acquires complex poles, it can enhance the power of certain frequency components as a resonator. According to the frequency response in Fig. 7(d), we can see the boosted frequency component near the 400th frequency bin. For F0 of read speech, the ripple may be unnecessary. However, this property of `AR-RMDN` can be used for modeling the vibrato in singing voice [24].

4.4 Discussion

The extended AR-RMDN allows the inverse AR filter to have complex poles. However, it also puts strong assumption on the location of the poles. For example, a filter with more than one real pole and one pair of complex poles is not supported. Besides, as the filter's parameters are learned through back-propagation, the initial value should be different for different sub-filter. Otherwise, all sub-filters will acquire the poles at the same location in each iteration of the training. Currently, we use a random Gaussian noise to initialize $\hat{\beta}_k$ and $\hat{\alpha}_k$. But it needs further investigation on the initial value for the parameters.

Because an ideal generative model is expected to generate natural data by drawing samples from the model, we also tested AR-RMDN's performance on this point. The sampling process

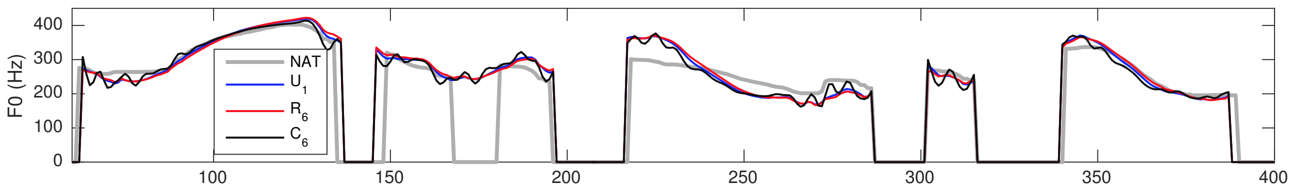

Fig. 7: F0 generated by AR-RMDN with different constraints in Table 2. The x-axis is the frame index and the frame shift is 5 ms. U_6 is not shown as the unstable inverse AR filter generated some very large and divergent F0 values.

was conducted in two steps: first, feature vectors $\boldsymbol{c}_{1:T}$ were sampled from the model $\prod_{t=1}^{T} p(\boldsymbol{c}_t; \mathcal{M}_t)$; second, acoustic feature vectors were computed by

$$\boldsymbol{o}_{1:T,d} = \boldsymbol{A}^{(d)-1}\boldsymbol{c}_{1:T,d}, \text{ where } d \in [1, D]. \tag{21}$$

However, the generated trajectories were still noisy and not perceptually convincing. One possible reason is that the AR model is not powerful enough in reproducing the temporal correlation by using linear transformation in Equation (21). Future work will look into the non-linear approach, e.g., moving the AR model to the hidden layers of the neural network.

4.5 Conclusion

This work first explained the AR-RMDN from the perspective of signal and filter. Based on the explanation, we found that the AR filter in a AR-RMDN learns to whiten the modulation spectrum of the acoustic feature trajectory in the training stage, and its inverse filter tries to de-whiten the modulation spectrum of the generated acoustic feature trajectory. This interpretation on the AR-RMDN shows its difference from the MS-based post-filtering approach, i.e., while the MS-based method directly manipulates the whole modulation spectrum towards the estimated statistics on natural data, the AR filter processes the data in 'time' domain and enhanced the energy of the low-frequency part of the modulation spectrum. Possibly due to that difference, the AR-RMDN generated better perceived speech in our experiments.

Originally, the implemented AR-RMDN can only acquire real poles for the inverse AR filter. Thus, the second part of this work extended AR-RMDN so that the inverse AR filter can have complex poles and be stable. This method simply constrains the parametric form of the filter's poles and works with the standard real-valued neural network. Experiments showed that this method is able to constrain the pole location accordingly. Although experiments on modeling F0 didn't show any improvement when the complex inverse AR filter is used, this method could be useful for other types of acoustic data with natural vibrato.

References

[1] Tokuda, K., Nankaku, Y., Toda, T., Zen, H., Yamagishi, J. and Oura, K.: Speech synthesis based on hidden Markov models, *Proceedings of the IEEE*, Vol. 101, No. 5, pp. 1234–1252 (2013).

[2] Ling, Z.-H., Kang, S.-Y., Zen, H., Senior, A., Schuster, M., Qian, X.-J., Meng, H. M. and Deng, L.: Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends, *IEEE Signal Processing Magazine*, Vol. 32, No. 3, pp. 35–52 (2015).

[3] Zen, H., Senior, A. and Schuster, M.: Statistical parametric speech synthesis using deep neural networks, *Proc. ICASSP*, pp. 7962–7966 (2013).

[4] Fan, Y., Qian, Y., Xie, F. and Soong, F. K.: TTS Synthesis with Bidirectional LSTM Based Recurrent Neural Networks, *Proc. INTERSPEECH*, pp. 1964–1968 (2014).

[5] Takaki, S. and Yamagishi, J.: A deep auto-encoder based low-dimensional feature extraction from FFT spectral envelopes for statistical parametric speech synthesis, *Proc. ICASSP*, pp. 5535–5539 (2016).

[6] Wang, X., Takaki, S. and Yamagishi, J.: Investigating Very Deep Highway Networks for Parametric Speech Synthesis, *Proc. SSW9*, pp. 181–186 (2016).

[7] Zen, H. and Senior, A.: Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis, *Proc. ICASSP*, pp. 3844–3848 (2014).

[8] Wang, X., Takaki, S. and Yamagishi, J.: An autoregressive recurrent mixture density network for parametric speech synthesis, *Proc. ICASSP*, p. to appear (2017).

[9] Keiichi, T., , Takayoshi, Y., Takashi, M., Takao, K. and Tadashi, K.: Speech parameter generation algorithms for HMM-based speech synthesis, *Proc. ICASSP*, pp. 936–939 (2000).

[10] Hashimoto, K., Oura, K., Nankaku, Y. and Tokuda, K.: The effect of neural networks in statistical parametric speech synthesis, *Proc. ICASSP*, pp. 4455–4459 (2015).

[11] Zen, H. and Sak, H.: Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis, *Proc. ICASSP*, pp. 4470–4474 (2015).

[12] Hashimoto, K., Oura, K., Nankaku, Y. and Tokuda, K.: Trajectory training considering global variance for speech synthesis based on neural networks, *Proc. ICASSP*, pp. 5600–5604 (2016).

[13] Toda, T. and Tokuda, K.: A speech parameter generation algorithm considering global variance for HMM-based speech synthesis, *IEICE Transactions on Information and Systems*, Vol. 90, No. 5, pp. 816–824 (2007).

[14] Takamichi, S., Toda, T., Neubig, G., Sakti, S. and Nakamura, S.: A postfilter to modify the modulation spectrum in HMM-based speech synthesis, *Proc. ICASSP*, pp. 290–294 (2014).

[15] Takamichi, S.: Acoustic modeling and speech parameter generation for high-quality statistical parametric speech synthesis, PhD Thesis, Nara Institute of Science and Technology (2016).

[16] Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks, PhD Thesis, Technische Universität München (2008).

[17] Bishop, C. M.: Mixture Density Networks, Technical report, Aston University (2004, http://eprints.aston.ac.uk/373/).

[18] Zen, H., Tokuda, K. and Kitamura, T.: Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences, *Computer Speech & Language*, Vol. 21, No. 1, pp. 153–173 (2007).

[19] Shannon, M., Zen, H. and Byrne, W.: Autoregressive models for statistical parametric speech synthesis, *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 21, No. 3, pp. 587–597 (2013).

[20] King, S. and Karaiskos, V.: The Blizzard Challenge 2011, *Proc. Blizzard Challenge 2011*, pp. 1–10 (2011).

[21] Kawahara, H., Masuda-Katsuse, I. and Cheveigne, A. d.: Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds, *Speech Communication*, Vol. 27, pp. 187–207 (1999).

[22] HTS Working Group: The English TTS System Flite+HTS_engine (2014).

[23] Weninger, F., Bergmann, J. and Schuller, B.: Introducing CURRENT: The Munich open-source CUDA recurrent neural network toolkit, *The Journal of Machine Learning Research*, Vol. 16, No. 1, pp. 547–551 (2015).

[24] Saitou, T., Goto, M., Unoki, M. and Akagi, M.: Vocal conversion from speaking voice to singing voice using STRAIGHT, *Proc. INTERSPEECH*, pp. 4005–4006 (2007).