10th ISCA Speech Synthesis Workshop
20-22 September 2019, Vienna, Austria

# Loss Function Considering Temporal Sequence for Feed-Forward Neural Network–Fundamental Frequency Case

*Noriyuki Matsunaga*[1, 2], *Yamato Ohtani*[1], *Tatsuya Hirahara*[2]

[1]AI, Inc.
[2]Toyama Prefectural University
matsunaga@ai-j.jp, ohtani@ai-j.jp, hirahara@pu-toyama.ac.jp

## Abstract

This paper describes a novel loss function for training feed-forward neural networks (FFNNs), which can generate smooth speech parameter sequences without post-processing. In statistical parametric speech synthesis based on deep neural networks (DNNs), maximum likelihood parameter generation (MLPG) or recurrent neural networks (RNNs) are generally used to generate smooth speech parameter sequences. However, because the MLPG process requires utterance-level processing, it is not suitable for speech synthesis requiring low latency. Furthermore, networks such as long short-term memory RNNs (LSTM-RNNs) have high computational costs. As RNNs are not recommended in limited computational resource situations, we look at employing FFNNs as an alternative. One limitation of FFNNs is that they train to ignore relationships between speech parameters in adjacent frames. To overcome this limitation and generate smooth speech parameter sequences from FFNNs alone, we propose a novel loss function that uses long- and short-term features from speech parameters. We evaluated the proposed loss function with a focus on the fundamental frequency (F0) at found that, using the proposed loss function, an FFNN-only approach can generate F0 contours that are perceptually equal to or better in terms of naturalness than those generated by MLPG or LSTM-RNNs.

**Index Terms**: long short-term, loss function, feed-forward neural network, low latency, fundamental frequency

## 1. Introduction

Statistical parametric speech synthesis based on deep neural networks (DNNs) has become a popular approach in recent years. DNNs such as WaveNet [2] and several generative adversarial networks [3] have enabled speech synthesis with higher quality than before. However, although computer performance has improved, the computational cost of DNNs remains an issue for low-latency speech synthesis. The lower the load of speech synthesis processing or the faster the speech synthesis processing, the more speech synthesis requests can be responded by one machine, and the operating cost of the speech synthesis service is reduced. Reducing power consumption in mobile devices by lowering the computational cost increases the available hours of device usage. In spoken dialog systems, the speech recognition and natural language generation processes run simultaneously with the speech synthesis process, meaning all computational resources cannot be used for speech synthesis. Under limited computational resources, it is also necessary to sequentially output speech waveforms while predicting speech parameters. For these reasons, the computational cost of DNNs must be reduced.

DNN-based statistical parametric speech synthesis systems use two major methods to model temporal sequences. The first is maximum likelihood parameter generation (MLPG) [4]. Because MLPG generates speech parameters by maximizing the likelihoods for the static and dynamic features of an entire utterance, DNNs must be used to predict the features before it is applied. This makes MLPG unsuitable for low-latency speech synthesis despite its ability to generate smooth speech parameters. The other method is recurrent neural networks (RNNs) [5], which generate speech parameters based on the relationships between the speech parameters in adjacent frames. Because RNNs perform recursive processing while retaining previous information, they do not require the post-processing for smoothing such as MLPG. Although RNNs can also generate smooth speech parameters, they have a non-negligible computational cost. In particular, the computational costs of long short-term memory RNNs (LSTM-RNNs), one of the most popular and highly performing types of RNNs, are high owing to their use of complex recursive structures.

Feed-forward neural networks (FFNNs) are basic DNNs with simplified structures that reduce computational costs. Although FFNNs are suitable for low-latency speech synthesis because they work on a frame-by-frame basis, they train to ignore relationships between speech parameters in adjacent frames. By overcoming this limitation, FFNN-only systems can be used to generate smooth speech parameter sequences.

In this paper, we propose an FFNN-only system to generate smooth speech parameter sequences based on a novel loss function that can be used to extract both long- and short-term features. As the fundamental frequency (f0) contour is crucial in achieving naturalness in synthesized Japanese speech, we focus on F0 contour generation using the loss function.

## 2. Proposed loss function

We define the linguistic feature sequence $\boldsymbol{x} = [\boldsymbol{x}_1^\top, \cdots, \boldsymbol{x}_t^\top, \cdots, \boldsymbol{x}_T^\top]^\top$ that includes auxiliary features $\boldsymbol{y} = [\boldsymbol{y}_1^\top, \cdots, \boldsymbol{y}_t^\top, \cdots, \boldsymbol{y}_T^\top]^\top$ as a natural speech parameter sequence and $\hat{\boldsymbol{y}} = [\hat{\boldsymbol{y}}_1^\top, \cdots, \hat{\boldsymbol{y}}_t^\top, \cdots, \hat{\boldsymbol{y}}_T^\top]^\top$ as a DNN-generated speech parameter sequence, where $t$ and $T$ are a frame index and the total frame length, respectively; $\boldsymbol{x}_t = \left[x_t^{(1)}, \cdots, x_t^{(k)}, \cdots, x_t^{(K)}\right]$ and $\boldsymbol{y}_t = \left[y_t^{(1)}, \cdots, y_t^{(d)}, \cdots, y_t^{(D)}\right]$ are the linguistic parameter vector and speech parameter vector at frame $t$, respectively; $k$ and $K$ are an index and the length of the linguistic feature vector, respectively; and $d$ and $D$ are an index and the length of the speech parameter vector, respectively;

Using the proposed loss function, a sequence of short-term segments $\boldsymbol{X} = [\boldsymbol{X}_1, \cdots, \boldsymbol{X}_t, \cdots, \boldsymbol{X}_T]$ and $\boldsymbol{Y} = [\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_t, \cdots, \boldsymbol{Y}_T]$ are formed by splitting $\boldsymbol{x}$ and $\boldsymbol{y}$ into short term $[t + L, t + R]$,


10.21437/SSW.2019-26

respectively, where $\boldsymbol{Y}_t = [\boldsymbol{y}_{t+L}^\top, \cdots, \boldsymbol{y}_{t+\tau}^\top, \cdots, \boldsymbol{y}_{t+R}^\top]^\top$ is a short-term segment at frame $t$, $L \leq 0$ is a backward lookup frame count, $R \geq 0$ is a forward lookup frame count, and $\tau$ is a short-term lookup frame index ($L \leq \tau \leq R$). In an FFNN, a sequence $\hat{\boldsymbol{y}}_{t+\tau}$ corresponding to $\boldsymbol{x}_{t+\tau}$ is independently predicted regardless of the adjacent frames. We introduce loss functions of time-domain constraint (TD) and local variance (LV) to relate adjacent frames in $\boldsymbol{Y}_t$ that propagate over the long-term frames during the training phase because $\boldsymbol{Y}_t$ and $\boldsymbol{Y}_{t+\tau}$ overlap. Under the proposed loss function, the explicitly defined short-term relationships between speech parameters implicitly propagate to the long term, which allows the FFNNs to train temporary sequences in a manner like LSTM-RNNs.

The proposed loss function is given by the weighted summation of the outputs of the loss functions as follows:

$$\mathcal{L}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \sum_i \omega_i \mathcal{L}_i(\boldsymbol{Y}, \hat{\boldsymbol{Y}}), \tag{1}$$

where $i = \{\mathrm{TD}, \mathrm{LV}, \mathrm{GV}\}$ is the loss function identifier and $\omega_i$ is the weight of the loss function of identifier $i$.

### 2.1. Time-domain constraint loss function

$\boldsymbol{Y}_{\mathrm{TD}} = [\boldsymbol{Y}_1^\top \boldsymbol{W}, \cdots, \boldsymbol{Y}_t^\top \boldsymbol{W}, \cdots, \boldsymbol{Y}_T^\top \boldsymbol{W}]$ is a sequence in a $D \times M$ matrix of time-domain constraint features at each short-term $[t+L, t+R]$. The time-domain constraint loss function $\mathcal{L}_{\mathrm{TD}}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ is defined as the mean squared error of the difference between $\boldsymbol{Y}_{\mathrm{TD}}$ and $\hat{\boldsymbol{Y}}_{\mathrm{TD}}$ as follows:

$$\mathcal{L}_{\mathrm{TD}}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \frac{1}{TMD} \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{d=1}^{D} \left(\boldsymbol{Y}_{\mathrm{TD}} - \hat{\boldsymbol{Y}}_{\mathrm{TD}}\right)^2, \tag{2}$$

where $\boldsymbol{W} = [\boldsymbol{W}_1^\top, \cdots, \boldsymbol{W}_m^\top, \cdots, \boldsymbol{W}_M^\top]$ is a $(-\mathrm{L} + \mathrm{R} + 1) \times M$ coefficient matrix that relates adjacent frames in the short-term $[t+L, t+R]$, $\boldsymbol{W}_m = [w_{mL}, \cdots, w_{m0}, \cdots, w_{mR}]$ is a coefficient vector, and $m$ and $M$ are an index and the total number of coefficient vectors, respectively.

To replicate the functionality of unidirectional RNNs, we set $L$ to be less than or equal to –1 and $R$ to 0. $\boldsymbol{W} = [\boldsymbol{W}_1^\top, \boldsymbol{W}_2^\top]$ comprises two coefficient vectors: $\boldsymbol{W}_1 = [0, \cdots, 0, 0, w_1]$ for obtaining static feature at frame $t$, and $\boldsymbol{W}_2 = [0, \cdots, 0, -w_2, w_2]$ for obtaining the delta between frames $t$ and $t-1$. $\boldsymbol{W}_2$ is introduced because relative changes in F0 are deeply related to Japanese accent perception [6]. Focusing on $\boldsymbol{Y}_{\mathrm{TD}} - \hat{\boldsymbol{Y}}_{\mathrm{TD}}$ at frame $t$ in (2), the equation can be rearranged when the loss is assumed to be zero to obtain the following recurrence formula:

$$\hat{\boldsymbol{y}}_t = \boldsymbol{y}_t - \frac{w_2}{w_1 + w_2} \boldsymbol{y}_{t-1} + \frac{w_2}{w_1 + w_2} \hat{\boldsymbol{y}}_{t-1}. \tag{3}$$

Because the loss function considers this recurrence formula, setting the values of $\boldsymbol{W}$ as described above enables recursive training in the manner of an RNN.

### 2.2. Local variance loss function

$\boldsymbol{Y}_{\mathrm{LV}} = [\boldsymbol{v}_1^\top, \cdots, \boldsymbol{v}_d^\top, \cdots, \boldsymbol{v}_T^\top]^\top$ is a sequence of variances in the short-term $[t+L, t+R]$. The local variance [7] loss function $\mathcal{L}_{\mathrm{LV}}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ can be defined as the mean absolute error of the difference between $\boldsymbol{Y}_{\mathrm{LV}}$ and $\hat{\boldsymbol{Y}}_{\mathrm{LV}}$ as follows:

$$\mathcal{L}_{\mathrm{LV}}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \frac{1}{TD} \sum_{t=1}^{T} \sum_{d=1}^{D} \left| \boldsymbol{Y}_{\mathrm{LV}} - \hat{\boldsymbol{Y}}_{\mathrm{LV}} \right|. \tag{4}$$

Here, $\boldsymbol{v}_t = [v_{t1}, \cdots, v_{td}, \cdots, v_{tD}]$ is a $D$-dimensional variance vector at frame $t$ and $v_{td}$ is the $d^{\mathrm{th}}$ variance at frame $t$ given as

$$v_{td} = \frac{1}{-L + R + 1} \sum_{\tau=L}^{R} \left(y_{(t+\tau)d} - \bar{y}_{td}\right)^2, \tag{5}$$

where $\bar{y}_{td}$ is the $d^{\mathrm{th}}$ mean at frame $t$ given as

$$\bar{y}_{td} = \frac{1}{-L + R + 1} \sum_{\tau=L}^{R} y_{(t+\tau)d}. \tag{6}$$

### 2.3. Global variance loss function

The global variance [8] loss function $\mathcal{L}_{\mathrm{GV}}(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ is defined as the mean absolute error of the difference between $\boldsymbol{Y}_{\mathrm{GV}} = [V_1, \cdots, V_d, \cdots, V_D]$, a $D$-dimensional variance vector of $\boldsymbol{y} = \boldsymbol{Y}|_{\tau=0}$, and $\hat{\boldsymbol{Y}}_{\mathrm{GV}}$ as follows:

$$\mathcal{L}_{\mathrm{GV}}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \frac{1}{D} \left| \boldsymbol{Y}_{\mathrm{GV}} - \hat{\boldsymbol{Y}}_{\mathrm{GV}} \right|, \tag{7}$$

where $V_d$ is the $d^{\mathrm{th}}$ variance given as

$$V_{\boldsymbol{d}} = \frac{1}{T} \sum_{t=1}^{T} (y_{td} - \bar{y}_d)^2, \tag{8}$$

where $\bar{y}_d$ is the $d^{\mathrm{th}}$ mean given as

$$\bar{y}_d = \frac{1}{T} \sum_{t=1}^{T} y_{td}. \tag{9}$$

## 3. Experimental conditions

Japanese speech data obtained from a female professional speaker were extracted into training and test data sets comprising 2,000 and 100 utterances, respectively.

From the speech data and associated transcriptions, phonetic alignments were manually refined by skilled labelers following automatic generation. The frame-level linguistic features for the DNN included 517 linguistic contexts based on the Japanese instantiation of the HMM/DNN-based Speech Synthesis System (e.g., phoneme identifiers, place of articulation codes, number of morae in an accent phrase, position of the mora in an accent phrase, position of the current frame in the current phoneme, etc.) [9]. Logarithmic F0 (log F0) values were extracted every 5 ms from the speech data sampled at 48 kHz and 16 bits using the WORLD system [10]. To model log F0 sequences, silent and unvoiced frames were interpolated. Both input and output features were normalized in advance, with the input features subjected to a robust normalization method to remove outliers [11] and the output features normalized to z-scores based on the global mean and variance calculated from the training data set.

Table 1 lists the architectural features of the DNNs used in the experiment. The number of parameters of each DNN was set to be almost the same as in architecture **#1**. Each DNN was trained by randomly selecting the training data over 20 epochs and an utterance-level batch size and removing the silent frames from the training data.

Architecture of model **#1** is the proposed training method. It models log F0 sequences by directly applying the FFNNs with the proposed loss function with optimum settings determined by a grid search after the effects of each loss function have been checked independently.

Architecture of model **#2** is the most basic training method [1], in which static and dynamic log F0 feature sequences are modeled by FFNNs using the mean squared error (MSE) criterion, and MLPG is applied to predicted feature sequences.

Table 1: *Architecture of DNNs. $\Delta^0$ and $\Delta^1$ denote static and dynamic features, respectively.*

| **Model no.** | **#1** | **#2** | **#3** | **#4** |
|---|---|---|---|---|
| **Network** | FFNN | FFNN | FFNN | LSTM-RNN |
| **Input layer** | 517 nodes | 517 nodes | 517 nodes | 517 nodes<br>25 time-steps |
| **Hidden layer** | ReLU<br>512 nodes<br>4 layers | ReLU<br>512 nodes<br>4 layers | ReLU<br>512 nodes<br>4 layers | LSTM<br>320 nodes<br>1 layer |
| **Output layer** | Linear<br>1 node ($\Delta^0$) | Linear<br>2 nodes ($\Delta^0, \Delta^1$) | Mean / Variance<br>Linear / Softplus<br>2 nodes / 2 nodes<br>($\Delta^0, \Delta^1$) / ($\Delta^0, \Delta^1$) | RNN<br>Linear<br>1 node ($\Delta^0$) |
| **Number of parameters** | 1,053,697 | 1,054,210 | 1,055,236 | 1,072,962 |
| **Loss function** | Proposed | MSE | Trajectory | MSE |
| **Optimizer** | Adam | Adam | Adam | Adam |
| **Post - processing** | None | MLPG with variance from training dataset | MLPG | None |

The variances of the static and dynamic features used in MLPG are calculated from the training data set, and their values remain the same in all frames.

Architecture of model **#3** is an advanced training method version of model **#2** in which static and dynamic log F0 feature sequences are modeled using FFNNs applying the minimum generation error (MGE) criterion [12, 13], and MLPG is applied to predicted feature sequences. Two output layers are used to obtain the means and variances of the static and dynamic features, respectively. To ensure that the variance is greater than zero, Softplus is used as the activating function of the output layer for variance.

Architecture of model **#4** is a training method that uses RNNs to eliminate MLPG post-processing. In this approach, log F0 sequences are directly modeled by LSTM-RNNs using the MSE criterion [5]. The settings of the LSTM-RNNs are determined by comparing their prediction errors with one hidden layer with 320 nodes, two hidden layers with 220 nodes per layer, three hidden layers with 200 nodes per layer, and four hidden layers with 180 nodes per layer. The timestep is set to 25 frames, corresponding to 126 ms of the average mora length, as calculated from the training data set.

One hundred utterances not included in the training data set were used for evaluation. To objectively evaluate the predicted F0 sequences, the absolute error for each frame of F0 sequences $E_{\mathbf{y}}$, the absolute error of the utterance-level standard deviation $E_{\mathrm{SD}}$, and the mean absolute error of the power spectrum $E_{\mathrm{R}}$ obtained from a section of 128 frames centered on frame $t$ were calculated and drawn using box plots. As high-frequency components increase with the occurrence of discontinuity in an F0 sequence, $E_{\mathrm{R}}$ serves as an index of the roughness of the sequence.

Seven listeners participated in paired comparison tests applying the comparison category rating method to evaluate the naturalness of synthesized speech prosody [14]. Synthesized speeches were generated using predicted F0 sequences with spectra, aperiodicity ratios, and durations matching those of natural speech. In the paired comparison tests, the listeners were asked to rate the prosody of the second stimulus on a scale of 1 to 6 (Table 2). The listeners were instructed to focus on the accent and intonation when listening to the stimulus pairs. No feedback was provided to any of the listeners.

Table 2: *Evaluation categories and scores measuring prosody of synthesized speech segments in paired comparison test. "A" and "B" denote the first and second stimuli, respectively.*

| **No.** | **Category**<br>The prosody of "B" compared to that of "A" is | **Score**<br>"A" | **Score**<br>"B" |
|---|---|---|---|
| **1** | Clearly better | 0 | 2 |
| **2** | Better | 0 | 1 |
| **3** | Almost as good | 1 | 1 |
| **4** | Almost as bad | 0 | 0 |
| **5** | Worse | 1 | 0 |
| **6** | Clearly worse | 2 | 0 |

# 4. Experimental results

### 4.1. Effect of time-domain constraint loss function

The first row in Figure 1 shows the effect of the time-domain constraint loss function as $w_2$ is varied from 1, 5, 10, to 15 with $L = -1$, R = 0, $w_1 = 1$, $\omega_{\mathrm{TD}} = 1$, $\omega_{\mathrm{LV}} = 0$, and $\omega_{\mathrm{GV}} = 0$. As $w_2$ increases, $E_{\mathrm{R}}$ decreases, and the F0 sequences become smoother, although at the same time, $E_{\mathrm{SD}}$ increases slightly, and the F0 sequence become slightly monotonous. To obtain a smooth F0 sequence, it is necessary to focus on the delta feature.

### 4.2. Effect of local variance loss function

The second row in Figure 1 shows the effect of the local variance loss function as $\omega_{\mathrm{LV}}$ is varied from 1, 5, to 10 with $L = -15$, R = 0, $w_1 = 1$, $w_2 = 1$, $\omega_{\mathrm{TD}} = 1$, and $\omega_{\mathrm{GV}} = 0$. We set $w_2$ to 1 to confirm that this loss function has the effect of smoothing the F0 sequence. Although this loss function decreases $E_{\mathrm{R}}$ slightly, the F0 sequence remains rough. This loss function does not exhibit enough effect on its own, but it does

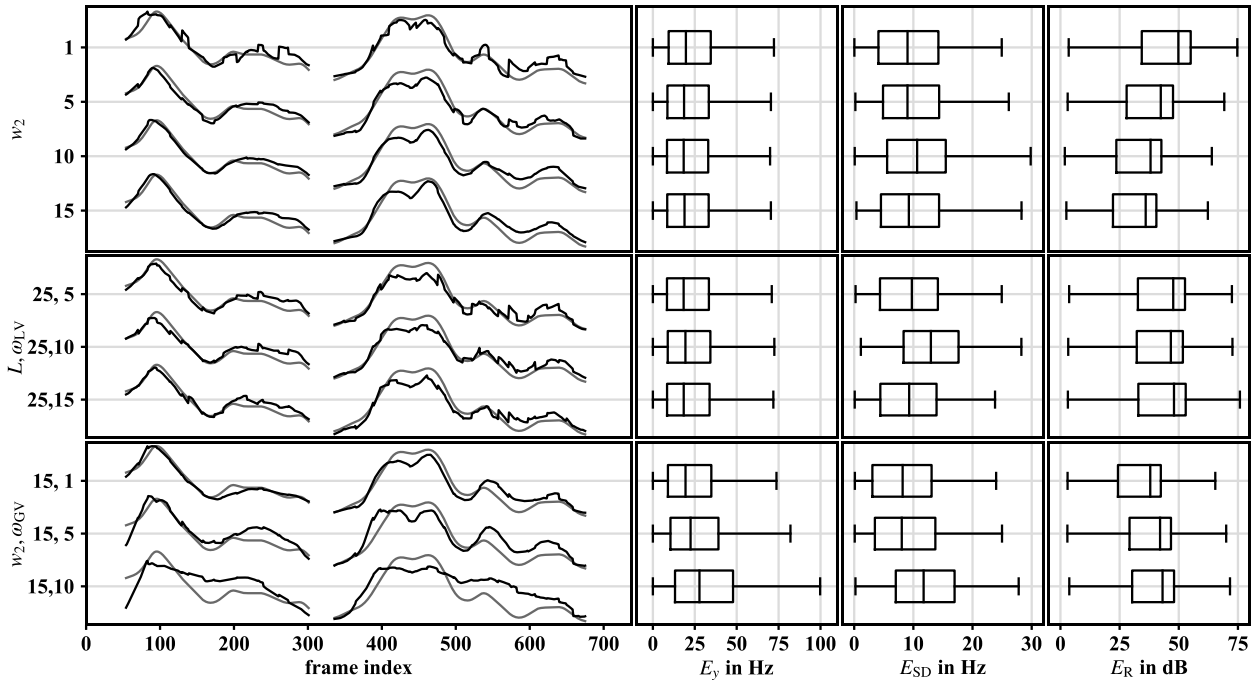


Figure 1: *Effects of loss functions within proposed loss function: time-domain constraint loss function (1st row), local variance loss function (2nd row), and global variance loss function (3rd row). The 1st column represents samples of predicted F0 sequence; 2nd column represents box plots of the absolute error of F0 sequence; 3rd column represents box plots of the absolute error of utterance-level standard deviation; 4th column represents $E_R$, roughness of F0 sequence. In the 1st column, the black and dark gray solid curves are predicted F0 and target F0 sequences, respectively.*

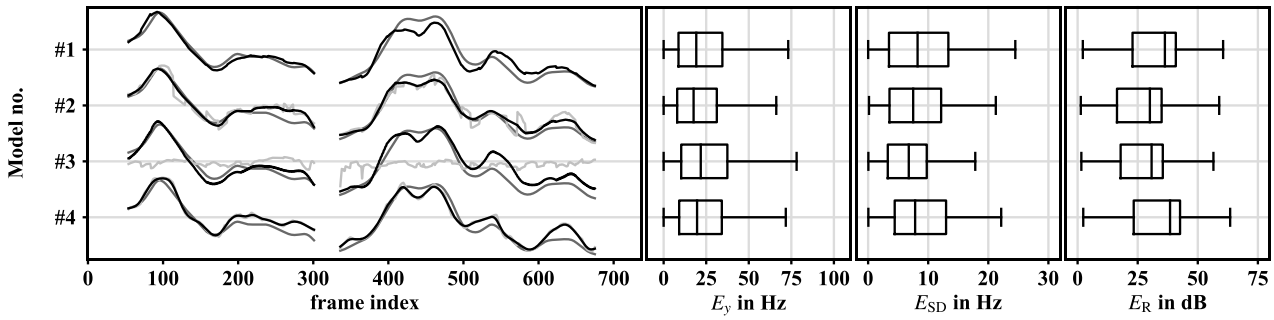


Figure 2: *Comparison of models: FFNNs with the proposed loss function (**#1**), FFNNs with MSE criterion (**#2**), FFNNs with MGE criterion (**#3**), and LSTM-RNNs with MSE (**#4**). The order of the columns is the same as in Figure 1. In the 1st column, the black, light gray, and dark gray solid curves are predicted F0 sequence after post-processing, predicted F0 sequence before post-processing, and target F0 sequence, respectively.*

serve as a weak constraint on the time-domain constraint and global variance loss functions.

### 4.3. Effect of global variance loss function

The third row in Figure 1 shows the effect of the global variance loss function as $\omega_{GV}$ is varied from 1, 5, to 10 with $L = -1$, $R = 0$, $w_1 = 1$, $w_2 = 15$, $\omega_{TD} = 1$, and $\omega_{LV} = 0$. At values of $\omega_{GV}$ below a specific fraction of $w_2$, the F0 sequence has a completely smooth global variance; as $\omega_{GV}$ increases beyond this value, discontinuities in the F0 sequence began to gradually appear, and, eventually, the sequence collapses.

### 4.4. Comparison by objective evaluation

Figure 2 shows the results of an objective evaluation of the proposed loss function. As grid search results, the parameters of the loss function are $L = -15$, $R = 0$, $w_1 = 1$, $w_2 = 20$, $\omega_{TD} = 1$, $\omega_{LV} = 1$, and $\omega_{GV} = 1$. All F0 sequences are smooth; in particular, those sequences produced using MLPG are smoother than those produced using the proposed loss function or LSTM-RNNs. The values of $E_{\boldsymbol{y}}$, $E_{SD}$, and $E_R$ produced using model **#1** are similar to those produced using model **#4**, confirming that the proposed loss function can simulate the LSTM-RNN mechanism. A comparison of the black solid and light gray curves produced by model **#2** and model **#3** in the first column reveals that MLPG was required because the F0 sequences were discontinuous or collapsed before post-processing. In particular, under model **#3**, the pre-post-processing F0 sequences were nearly the same as the global mean, while those obtained from post-processing were primarily generated from the dynamic feature.

### 4.5. Comparison by subjective evaluation

Table 3: *Percentage of chosen category when the stimuli of **#1** are summed as the first stimulus "A".*

| Category no. | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **#1** vs. **#2** | 0.0 | 4.0 | 83.3 | 1.7 | 9.7 | 1.3 |
| **#1** vs. **#3** | 6.7 | 24.7 | 51.1 | 2.0 | 15.1 | 0.3 |
| **#1** vs. **#4** | 8.3 | 39.1 | 45.4 | 1.3 | 5.0 | 0.9 |

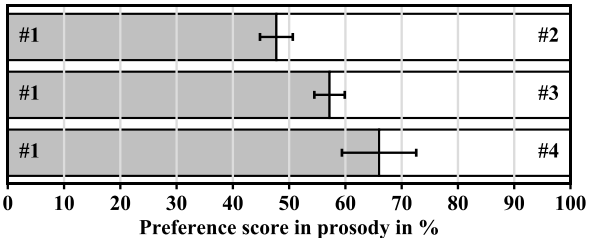


Figure 3: *Preference scores of paired comparison test on prosody quality with 95% confidence intervals.*

Table 4: *Average processing time in 100 times of prediction and post-processing when using a 1000-frame sequence.*

| Computing environment | #1 | #2 | #3 | #4 |
|---|---|---|---|---|
| CPU (1 thread) *1 | 131 | 680 | 682 | 1858 |
| CPU (4 threads) *1 | 51 | 617 | 609 | 1437 |
| CPU (1 thread) *2 | 78 | 134 | 131 | 1793 |
| CPU (8 threads) *2 | 23 | 78 | 76 | 814 |
| CPU (1 thread) / GPU *3 | 9 | 232 | 237 | 94 |
| CPU (8 threads) / GPU *3 | 8 | 230 | 235 | 94 |
| CPU (1 thread) / GPU *4 | 8 | 208 | 216 | 86 |
| CPU (16 threads) / GPU *4 | 8 | 215 | 221 | 83 |

[ms]

*1 Intel Core 2 Quad Q9550
*2 Intel Core i7 7700
*3 Intel Core i7 6700K and NVIDIA GeForce GTX1080
*4 Intel Core i9 9900K and NVIDIA GeForce RTX 2080

Table 3 and Figure 3 show the results of the subjective evaluation. The table shows the percentages of the chosen category obtained when the speeches synthesized by model **#1** are summed as the first stimulus. The figure shows a comparison of the preference scores among models **#1** to **#4**. All scores differ at a significance level of 0.1% by paired t-test, and the preference score of model **#1** is greater than those of models **#3** and **#4**, but lower (47.7%) than that of model **#2** (52.3%). However, more than 80% of the responses are of category 3, suggesting that there is no difference in quality between the prosodies of models **#1** and **#2**.

### 4.6. Comparison of processing time

Table 4 shows average processing times in 100 times of prediction and post-processing when using a 1000-frame sequence. Model **#1** was the fastest of all models in all computing environments. Models **#2** and **#3** were more than twice as slow as model **#1**. Model **#4** was the slowest of all models in the CPU only computing environment, however, model **#4** was as fast as model **#1** in computing environment with the GPU.

The post-processing MLPG used in models **#2** and **#3** was executed by the CPU, but in the computing environment with the GPU, even if the number of threads was changed, the processing time had plateaued about 200 ms.

## 5. Discussion

The perceptual quality of the predicted F0 sequence by model **#1** was better than that by models **#3** and **#4**, while $E_{\mathrm{y}}$ and $E_{\mathrm{SD}}$ of model **#1** were lager than those of other models. This is because models **#1** and **#2** explicitly train delta F0, which is important for perception of Japanese accents. In speech synthesis, obtaining better perceptual quality of the synthesized speech is more important than reducing the error of the predicted parameter sequence.

In consideration of the user experience, a system that has a lag before playing the synthesized speech is not good. In the system using model **#2** or **#3**, which require the utterance-level batch post-processing, the lag occurs since speech waveform is generated only after speech parameters of all frames have been generated. On the other hand, in the system using **#1** or **#4** will not generate lags since speech waveform can be generated while generating speech parameters.

The system using model **#4** generates almost no lag. Longer processing time, however, is necessary in poor computing environments such as embedded device or CPU with few cores and low clock speed, as shown in Table 4. That is, it is impossible to generate a real-time speech waveform.

Using the ordinary loss function in the DNN training phase enables the model architecture, model and optimizer parameters to be adjusted even when the training result is not good. Because it is hard to understand how they affect the output features, intuitively adjusting these parameters is not possible. By contrast, using the proposed loss function in the DNN training phase enables the adjustment of parameters, such as $L$, $w_1$, and $\omega_{\mathrm{LV}}$, related to the secondary features and extracted from the output feature. These parameters directly relate to the output feature, and adjusting them intuitively is possible.

An FFNN training method involving singing voice synthesis that does not require MLPG has been proposed [15]. The training method proposed in this paper is similar to this approach, but it does not require a convolutional neural network and diagonal covariance, and it can adjust training settings directly.

## 6. Conclusions

In this paper, we proposed a novel loss function based on long- and short-term features extracted from speech parameters using FFNNs. Subjective evaluation results showed that an FFNN using the proposed loss function can generate F0 contours that are perceptually equal to or better in terms of prosody naturalness than those generated by MLPG or LSTM-RNNs. The proposed loss function makes it possible to achieve DNN-based low-latency speech synthesis with limited computational resources.

## 7. References

[1] H. Zen, A. Senior, and M. Schuster, "Statistical Parametric Speech Synthesis Using Deep Neural Network," Proc. ICASSP, pp. 7962-7966, 2013.

[2] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: a generative model for raw audio," arXiv.1609.03499, 2016.

[3] Y. Saito, S. Takamichi, and H. Saruwatari, "Statistical Parametric Speech Synthesis Incorporating Generative Adversarial Networks," IEEE/ACM Trans. vol. 26, no. 1, 2018.

[4] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, K. Oura. “Speech synthesis based on hidden Markov models,” Proc. IEEE, Vol. 101, No. 5, pp. 1234-1252, May 2013.
[5] H. Zen, and H. Sak, “Unidirectional Long Short-Term Memory Recurrent Neural Network with Recurrent Output Layer for Low-Latency Speech Synthesis,” Proc. ICASSP, pp. 4470-4474, 2015.
[6] T. C. Ishi, N. Minematsu, and K. Hirose, “Identification of Japanese accent in continuous speech considering pitch perception,” IEICE Technical Report SP., 101(270), pp. 23-30, 2001.
[7] T. Nose, V. Chunwijitra, and T. Kobayashi, “A parameter Generatioin Algorithm Using Local Variance for HMM-Based Speech Synthesis,” Proc. IEEE, vol. 8, no. 2, pp. 221-228, 2014.
[8] T. Toda and K. Tokuda, “Speech Parameter Generation Algorithm Considering Global Variance for HMM-Based Speech Synthesis,” Proc. INTERSPEECH 2005, pp. 2801-2804, Lisbon, Portugal, September, 2005.
[9] HTS, http://hts.sp.nitech.ac.jp/
[10] M. Morise, F. Yokomori, and K. Osawa, “WORLD: a vocoder-based high-quality speech synthesis system for real-time applications,” Trans. IEICE, vol. E99-D, no. 7, pp. 1877-1884, 2016.
[11] N. Matsunaga, Y. Ohtani, and T. Hirahara, “Normalized method of linguistic feature for fundamental frequency of Japanese speech synthesis using deep neural network,” Proc. ASJ, 1-P-21, pp. 1089-1090, March 2019.
[12] Z. Wu, S. King, “Minimum trajectory error training for deep neural networks combined with stacked bottleneck features”, Proc. INTERSPEECH 2015, pp. 309-313, 2015.
[13] K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, “Trajectory training considering global variance for speech synthesis based on neural networks,” in Proc. Int. Conf. Acoust., Speech, Signal Process., pp. 5600-5604, Shanghai, China, Mar. 2016.
[14] ITU-T Recommendation P.800: Methods for subjective determination of transmission quality, https://www.itu.int/rec/T-REC-P.800-199608-I/en
[15] K. Nakamura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, “CNN-based speech parameter generation for singing voice synthesis,” Proc. ASJ, 2-10-4, pp. 1033-1034, March. 2019.