物联网平台广播系统的人工智能降噪算法研究

摘要： 本文围绕物联网广播系统中的语音降噪需求，重点对比分析了传统 WebRTC 降噪算法与基于 RNN（循环神经网络）的深度学习降噪算法的性能及应用场景。WebRTC 降噪算法基于经典的维纳滤波理论，依靠频谱估计实现动态噪声抑制。其优势在于计算复杂度低、实时性强，适合对延迟要求严格的场景。然而，该算法在复杂噪声环境下表现出一定局限性，尤其对频谱重叠的非语音噪声或多频混合噪声的处理存在不足，可能导致降噪效果欠佳或语音信号失真。RNN 降噪算法则利用深度学习结合时序数据建模的能力，通过对语音和噪声特征的精确提取，在复杂噪声环境中展现出更好的适应性。实验表明，RNN 在提升音质保真度和抑制背景噪声方面效果显著，尤其在人声与噪声频谱交叠的场景中，其降噪能力和听觉体验均优于 WebRTC。此外，RNN 还能够在人声静音时显著降低背景噪声，使广播系统整体音质更清晰自然。然而，RNN 对硬件计算性能需求较高，训练过程也依赖大量标注数据，适合对实时性要求较低且硬件资源充足的场景。

关键字：物联网广播系统；噪声抑制；WebRTC 算法；循环神经网络（RNN）；深度学习

Dynamic Switching Strategy of Base Station Based on Reinforcement Learning Prediction

Abstract: This paper focuses on the noise reduction requirements of IoT-based broadcasting systems and conducts a comparative analysis of the traditional WebRTC noise reduction algorithm and the RNN (Recurrent Neural Network)-based deep learning noise reduction algorithm. The WebRTC algorithm relies on classical Wiener filtering and spectral estimation to achieve dynamic noise suppression. Its advantages include low computational complexity and strong real-time performance, making it suitable for scenarios with strict latency requirements. However, it shows limitations in complex noise environments, particularly in handling overlapping non-speech noise or multi-frequency mixed noise, which may result in insufficient noise suppression or voice signal distortion.

In contrast, the RNN noise reduction algorithm leverages deep learning and temporal data modeling capabilities to extract speech and noise features more accurately. It demonstrates superior adaptability in complex noise environments. Experiments show that RNN outperforms WebRTC in improving audio quality and suppressing background noise, especially in scenarios where speech and noise spectra overlap. Additionally, RNN effectively reduces background noise during silent periods, providing a clearer and more natural audio experience for broadcast systems. However, RNN requires higher computational resources and relies on extensive labeled data for training, making it more suitable for scenarios with lower real-time demands and sufficient hardware support.

Key words: IoT Broadcasting Systems; Noise Reduction; WebRTC Algorithm; RNN (Recurrent Neural Network); Deep Learning.

0 引言

在本次课程当中，我们小组介绍了在嵌入式芯片与系统设计竞赛中设计的“宿舍智能交互/管理系统”，该系统基于高通 SOC 的物联网板卡开发，具有多种网络连接能力和推理加速能力，这使之具有了强大的边缘计算能力。该系统中有两个核心模块涉及了语音信号的采集与传输：视频通话模块、语音广播模块。原理上视频通话模块使用 WebRTC 协议中内置的降噪算法，而广播模块则只是简单的实现了音频的采集和广播。在实际的测试当中，WebRTC 协议内置的降噪算法使得视频通话质量较高，而未作任何处理的广播系统经常出现啸叫、回音、噪音问题。由于系统架构的独特设计，广播端的位置和设备可以灵活修改，我们难以使用构建广播室这种物理声学手段对广播效果进行优化。

与语音通话存在些许不同，广播场景并不存在多音源多终端带来的过多非线性问题，也没有实时通话要求的高实时性需求，但是存在广播音乐等对音质要求较高但与人声信号存在较大差异的问题，我们不一定仍要采用 WebRTC 中的降噪算法实现这个功能，本文将比较 WebRTC 中的噪音抑制算法与基于 RNN 的神经网络降噪算法之优劣，并通过测试给出基于 RNN 的降噪算法应用在该场景的优势。

本文研究是本课程相关先前系统的扩展，并不属于之前项目的成果，相关测试代码与用例详见：https://github.com/S3-110/GHT-NS-algorithm

1 WebRTC 中的降噪算法

本节大体介绍了 WebRTC 中基于维纳增益的降噪算法数学原理和代码实现，该算法具有计算简单、实时性强、语音失真小的特点。

1.1 整体架构

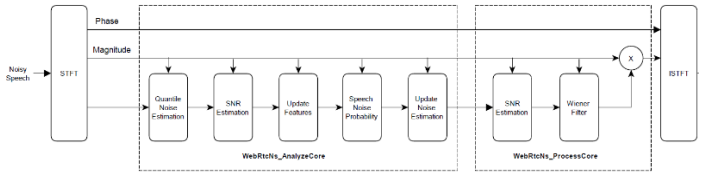

图 1 WebRTC NS 降噪模块流程图

如图所示，带有噪声的语音经过 STFT 模块后分解为幅度谱和相位谱，信号首先输入到 Analyze Core 进行估计和特征分析，首先对噪声进行量化估计，得到噪声的频谱特性，随后是计算降噪前的信噪比，随后提取并更新信号的特征(语音和噪声分布情况和能量情况)，得到特征后即可基于信号特征和历史数据，计算语音和噪声的概率分布，随后更新噪声估计模型。

在 Process Core 中，将执行降噪，使用维纳滤波器进行降噪，随后将处理后的幅度谱和相位谱结合，通过 ISTFT 后重建为降噪后的语音信号。

1.2 维纳滤波器

输入信号通过一个线性时不变系统之后产生一个输出信号，使得输出信号尽量逼近期望信号，使其估计误差最小化，能够最小化这个估计误差的最优滤波器称为维纳滤波器。

在语音降噪中，一般假设输入信号$y(n)$由干净语音$x(n)$和噪声信号$n(n)$组成:

$$y(n) = x(n) + n(n)$$

我们假设噪声和语音不相关，则得到：

$$R_{yy} = R_{xx} + R_{nn}$$

$$P_{yy} = P_{xx} + P_{nn}$$

$$R_{yx} = R_{xx}$$

$$P_{yx} = P_{xx}$$

可以得到最优的时频域滤波器估计可以分别表示为以下:

$$\mathbf{H}(\mathbf{w_k}) = \frac{P_{xx}(\mathbf{w_k})}{P_{xx}(\mathbf{w_k}) + P_{nn}(\mathbf{w_k})}$$

$$\mathbf{h} = \frac{R_{xx}}{R_{xx} + R_{nn}}$$

WebRTC 使用的维纳增益函数即:

$$G(k,l) = \frac{|X(k,l)|^2}{|Y(k,l)|^2} = 1 - \frac{|N(k,l)|^2}{|Y(k,l)|^2}$$

1.3 降噪代码解析[1]

```
// 噪声抑制主函数，输入为多频带语音信号，输出为处理后的语音信号
void WebRtcNs_ProcessCore(…) {
// 省略部分变量声明

// 如果多于一个频带，处理高频带
if (num_bands > 1) {
// 略…
}
// 计算加窗后的能量
Float energy1 = WindowingEnergy(self->window, self->dataBuf, self->anaLen, winData);
if (energy1 == 0.0) {
// 输入信号为零时，直接输出零信号
// 略过处理
return;
}
```

这里首先计算了输入信号的能量,如果为0(静音)直接输出0,避免了对静音信号的降噪处理而产生的异常效果。

```
// 对低频信号进行傅里叶变换
FFT(self, winData, self->anaLen, self->magnLen, real, imag, magn, NULL, 0, NULL, NULL);

// 计算维纳滤波器
ComputeDdBasedWienerFilter(self, magn, theFilter);

// 对信号进行频域滤波
for (i = 0; i < self->magnLen; i++) {
    theFilter[i] = fmaxf(fminf(theFilter[i], 1.f), self->denoiseBound); // 限制滤波器范围
    self->smooth[i] = theFilter[i];
    real[i] *= self->smooth[i];
    imag[i] *= self->smooth[i];
}

// 反傅里叶变换回时域
IFFT(self, real, imag, self->magnLen, self->anaLen, winData);
```

随后使用 FFT 算法将信号转换到了频域继续进行操作，根据相关信息计算出维纳滤波器参数，它根据信噪比（SNR）动态调整频域增益，实现维纳滤波的效果，滤波器值会被限制在 [denoiseBound, 1.0] 的范围内,避免对信号的过度抑制。

计算好参数后即将参数应用在信号频域上，通过 IFFT 将信号转换回到时域完成维纳滤波。

```
// 对输出信号进行缩放
float factor = 1.f; // 缩放因子
if (self->gainmap == 1 && self->blockInd > END_STARTUP_LONG) {
// 根据能量比计算缩放因子
    float energy2 = Energy(winData, self->anaLen);
    factor = sqrtf(energy2 / (energy1 + epsilon) + epsilon_squ);
}
```

最后根据语音概率调整了增益，如果处理的本段语音概率大

则增强信号，反之抑制噪声。

本节仅展示了核心的降噪模块的工作算法，实际使用的工程代码中还包括了 AGC 等音频增强方法，但由于降噪模块的性能总体依赖于此处理核心，本文在后续的测试当中直接使用整体降噪模块进行对比。

2 基于 RNN 的语音降噪算法

2.1 基于 RNN 语音降噪的原理

Jean-Marc Valin 等提出了一种用于噪声抑制的快速算法[2]，将传统数字信号处理与深度学习相结合，在降噪时需要精细调整的部分使用深度学习方法，而其他部分使用传统的 DSP 算法以降低算法的应用压力，这使得这个算法离线部署在我们的物联网系统中成为可能。

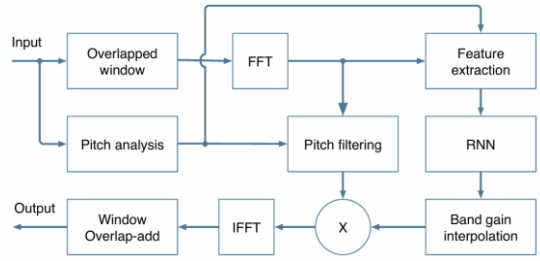

图 2 基于 RNN 算法的降噪系统结构

此种设计的系统结构主要结构(图 2)分为 Pitch analysis、Pitch filtering、Feature extraction、Band gain interpolation、RNN，下面将依次介绍非神经网络的部分，RNN 的结构将放到下一节介绍。

1) Pitch analysis

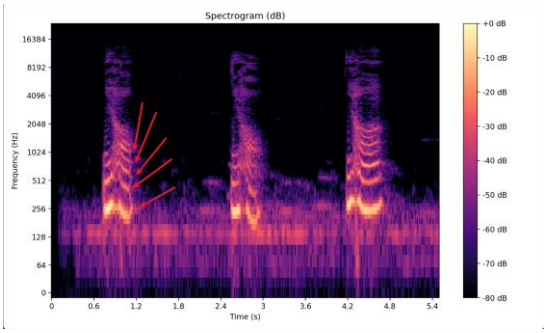

图 3 无噪的人声信号频谱图

Pitch Analysis 即音高分析，其作用是估计语音信号的音高(Pitch)，即基本频率(Fundamental Frequency，f_0）用于准确提取语音信号中的周期性特征。

人声信号来源于声带振动，这种振动很短的时间内振动的频率是比较平稳的，可以认为具有一个稳定的基频。我们从图 3 中即可明显看出，红色箭头指出了人声基频与其高次谐波（共振峰），这说明在时域上将出现较为明显的周期出现的幅度峰值，而音高分析就是要在时域找到这个基频，为后续人声增强和降噪做准备。

2) Pitch filtering

在从音高分析模块中得到了基频和谐波相关信息，通过 Pitch filtering 模块执行频域上的音高滤波，这对非周期性的噪声具有较强的抑制效果，可得到干净的人声信号，这样操作后的信号将用于后续的人声共振峰增强。

3) Feature extraction

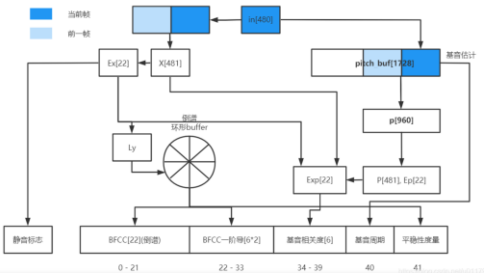

图 4 特征提取流程

Feature Extraction（特征提取） 是的主要目的是将原始信号中的信息压缩和转化为用于分析的特征表示。这些提取的特征保留了信号中的关键信息，同时降低了不必要的冗余数据量，从而使后续的任务更加高效和准确。在这里带上音高相关信息求取了 42 维的特征输入 RNN 网络，RNN 将给出 22 维参数用于后续操作。这里的详细操作详见图 4 给出的流程图，使用采样率为 48kHz 的音频时，每一帧的新数据长度变为 480，即图中的in[480]，将其与上一次输入的 480 个点数据拼接得到长度为 960 的数据，此为一帧。对其做长度为 960 的傅里叶变换并取前 481（代码中此值记作 FREQ_SIZE）点数据得到X。Ex则是利用X求出的 22 个频带的能量（使用三角滤波器，详见代码，本文略）。

4) Band gain interpolation

RNN 给出的参数仅有 22 维，通过插值将 22 维数据扩大到整个帧长点数的数据，这样就可以继续进行后续的 X 操作了。

2.2 RNN 神经网络架构

2.2.1 神经网络的构建

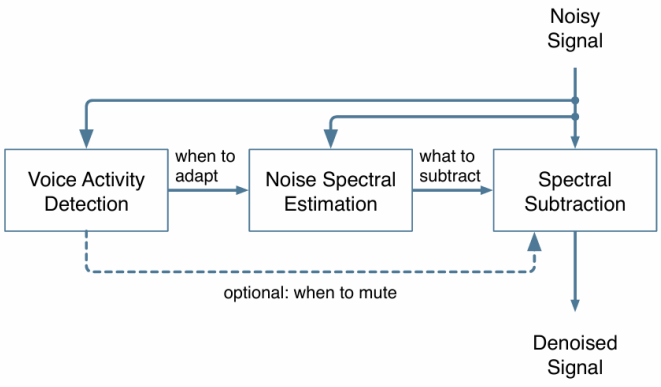

图 5 传统降噪算法结构

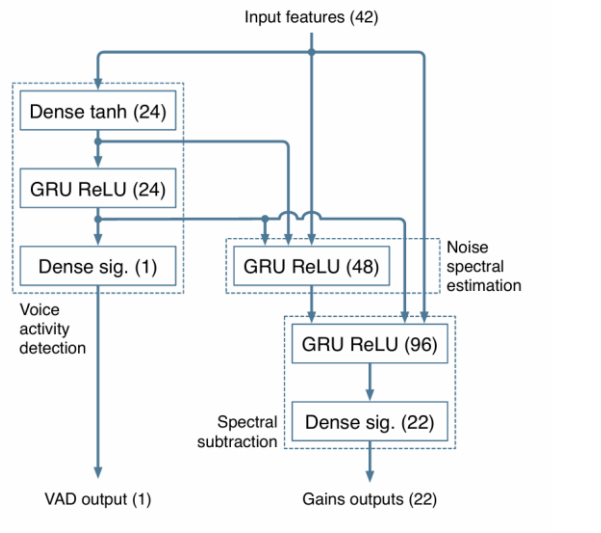

图 6 神经网络结构

这里的神经网络结构（图 6)与传统的噪声抑制算法结构（图 5）类似，都具有三个主要的构成部分（即语音检测、噪声估计、噪声削减，神经网络中构建为三个循环层）。这里采用的是 GRU 网络，原论文测试了 LSTM 和 GRU，发现 GRU 的效果要优于 LSTM。整个神经网络包含 4 层隐藏层，215 个单元，三个 GRU 层分别处理噪声抑制的核心任务

2.2.2 模型的训练优化

模型采用 McGill TSP 和 NTT 多语言数据库与多种现实场景的噪声音源（电风扇、汽车）融合形成多种信噪比的数据用于训练，其中静音或无信号频带的增益被标记为未定义，避免干扰训练。优化方面采用了增益平滑通过限制帧间衰减来提升输出的自然性。

$$g_b' = \max\left(g_b^{(\text{prev})} - \beta, g_b\right)$$

其中$g_b^{(\text{prev})}$是前一阵的增益，衰减因子 β=0.6 对应 135 毫秒的混响时间。

2.2.3 模型构建代码实现

训练出的模型具有 denoise_output 和 vad_output 两个核心输出，分别用于除去噪声和检测是否存在语音输出，接下来对模型的构建过程进行详细的介绍：

```
# 输入层，形状为 (时间步数，特征维度)
main_input = Input(shape=(None, 42), name='main_input')
tmp = Dense(24, activation='tanh', name='input_dense', kernel_constraint=constraint, bias_constraint=constraint)(main_input)
```

首先构建模型的第一层，输入层，输入的形状为（时间步数，特征维度），其中时间步数为 None，表示时间步数不固定，特征维度为 42。

接着构建特征嵌入层，使用全连接层将 42 维输入特征映射到 24 维，激活函数为 tanh。

```
# 语音活动检测 (VAD) 子网络
# 使用 GRU 提取时间序列特征，输出 24 维隐藏状态
vad_gru = GRU(24, activation='tanh', recurrent_activation='sigmoid', return_sequences=True,
name='vad_gru', kernel_regularizer=regularizers.l2(reg),kernel_constraint=constraint, recurrent_constraint=constraint,bias_constraint=constraint)(tmp)
# VAD 输出层，单神经元 sigmoid 激活，用于二分类（是否有语音活动）
vad_output = Dense(1, activation='sigmoid', name='vad_output'kernel_constraint=constraint, bias_constraint=constraint)(vad_gru)
```

随后的构建 VAD 子网络用于对输入的语音信号进行语音活动检测，输出一个二分类结果，即是否有语音活动，这将用于后期根据语音存在的状态对音频进行不同的增益/削减策略。

```
# 噪声特征提取子网络
# 将 VAD 隐藏状态、输入特征和特征嵌入层输出拼接
noise_input = concatenate([tmp, vad_gru, main_input])
# 使用 GRU 提取噪声特征，输出 48 维隐藏状态
noise_gru = GRU(48, activation='relu', recurrent_activation='sigmoid', return_sequences=True, name='noise_gru', kernel_regularizer=regularizers.l2(reg),
kernel_constraint=constraint, recurrent_constraint=constraint,bias_constraint=constraint)(noise_input)
```

接下来构建噪声特征提取子网络，将特征嵌入层输出、VAD 隐藏状态和输入特征拼接，使用 GRU 提取噪声特征，输出 48 维隐藏状态，此网络提供了噪声特征用于后续的语音降噪。

```
# 语音降噪子网络
# 将噪声特征、VAD 隐藏状态和输入特征拼接
denoise_input = concatenate([vad_gru, noise_gru, main_input])
# 使用 GRU 提取降噪特征，输出 96 维隐藏状态
denoise_gru = GRU(96, activation='tanh', recurrent_activation='sigmoid', return_sequences=True,name='denoise_gru', kernel_regularizer=regularizers.l2(reg), kernel_constraint=constraint, recurrent_constraint=constraint,
bias_constraint=constraint)(denoise_input)
# 降噪输出层，22 维向量，sigmoid 激活，对应频带增强系数
```

```
denoise_output = Dense(22, activation='sigmoid', name='denoise_output', kernel_constraint=constraint, bias_constraint=constraint)(denoise_gru)
```

最后的语音降噪子网络，将噪声特征、VAD 隐藏状态和输入特征拼接，使用 GRU 提取降噪特征，输出 96 维隐藏状态，降噪输出层输出 22 维向量，sigmoid 激活，对应频带增强系数。

```
# 构建模型
# 输入：main_input，输出：denoise_output（语音增强）和 vad_output（语音活动检测）
model = Model(inputs=main_input, outputs=[denoise_output, vad_output])
# 编译模型
model.compile(loss=['mean_squared_error', 'binary_crossentropy'],loss_weights=[10, 0.5], optimizer='adam')
```

模型使用的损失函数是均方误差和二元交叉熵，分别用于语音增强和语音活动检测，优化器使用 adam。

3 性能对比

3.1 测试数据集描述

测试数据集中人声语音采用 5s 无噪人声信号，叠加不同噪音使信噪比下降。

为了模拟实际的使用环境，我们采取风扇噪声与干净人声合成。

为了评估降噪能力，我们还采用白噪音与干净人声合成使信噪比达到 15db 来进行对比评估。

3.4 评估方法

本文采用的 RNN 降噪算法实现为 RNNoise 项目[3]的 C 语言实现[3]，WebRTC 中的降噪算法实现则是源于 Zhihan Gao 的 WebRTC-NS 的 C 语言实现**错误!未找到引用源。**。其中 RNNoise 的编译环境为 x86_64-linux-gnu，WebRTC-NS 的编译环境为 x86_64-w64-mingw32。

3.2 性能评估标准

为了客观评价不同降噪算法的降噪效果，我们采用直观观察频谱图对比和使用 speechmetrics[5]进行音质评估两种评估方式，其中 speechmetrics 是是一个用于测量语音信号质量的工具库。它封装了多种常见的语音信号质量客观评估指标，可以用来评估语音信号的清晰度、自然性、失真程度等。这些指标广泛用于语音处理领域，例如语音增强、语音识别、语音合成、语音压缩算法的性能评估。

3.3 降噪性能对比

3.2.1 干净人声频谱与加噪信号对比

图 7 展示了本次实验的干净人声频谱，可以观察到明显的人声信号特征，即存在基频和相应的高次谐波信号，并且存在代表音色的高频信号与较多的低频成分。

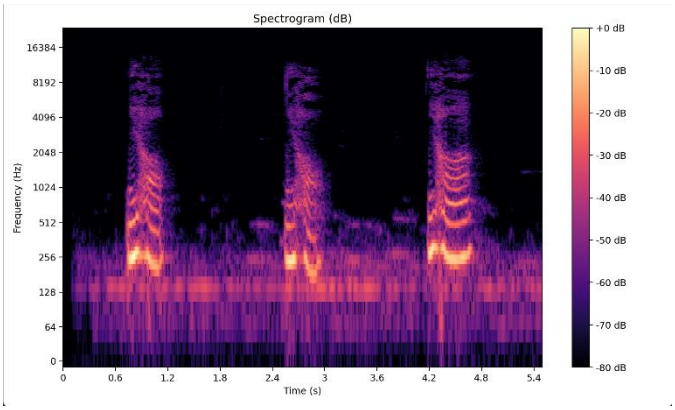

图 7 干净人声频谱

图 8 展示了本次实验的加高斯白噪声频谱，白噪声均匀分布在整个频域范围，方便在后续实验中对降噪效果进行直观对比。

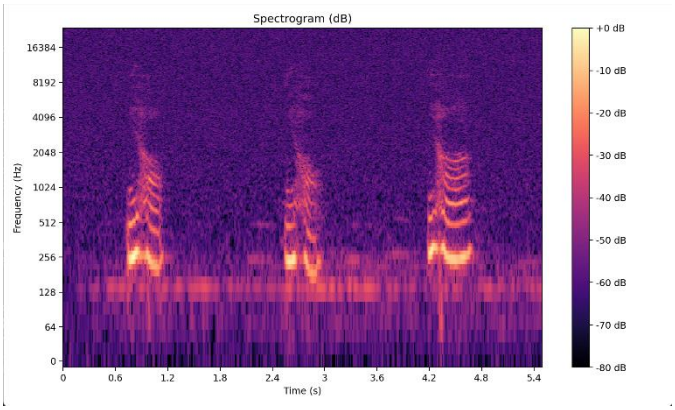

图 8 加高斯白噪声的频谱

图 9 展示了加风扇噪声的频谱，可以看到风扇噪声的频率也几乎占满了整个频域，但是与白噪声不同，风扇噪声存在一些谐波频率的功率峰值，且低频分布较多，类似粉红噪声。

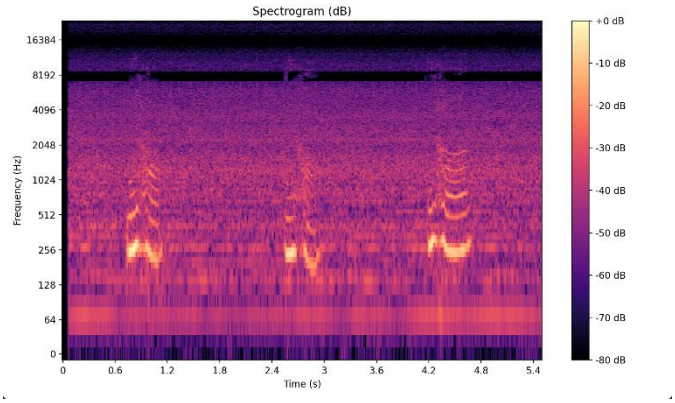

图 9 模拟办公环境噪声的频谱

3.2.1 白噪音降噪性能

FIR 滤波器作为较方便实现的简单数字滤波方案，可以有效

提高信号的 SNR，我们使用 0-2048Hz 的低通 FIR 滤波器，用以模拟我们当前采用的最简单数字滤波方案。可以看到该方案虽然提高了整体信号的信噪比，但是损失了全部高频信号，极大损伤了音质。

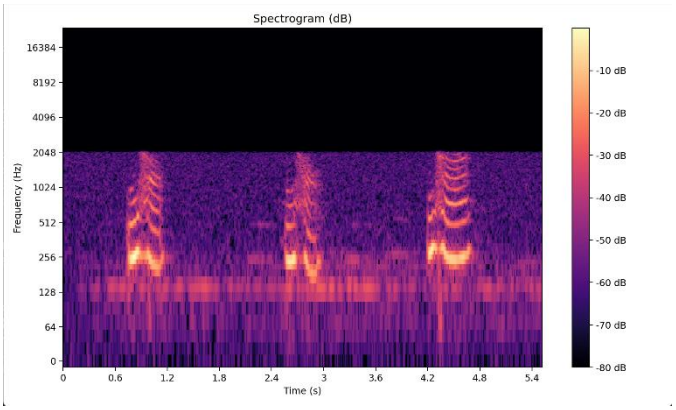

图 10 低通 FIR 滤波器降噪频谱

WebRTC 的降噪方案可以明显看出其对基频以及其谐波的增强，该方案也较好的检测出了有声区间和无声区间，无声的时候一直效果较好。该方案展现出了较好的对白噪声的滤波效果，且无需延时等待，该方案一直具有稳定的降噪效果。

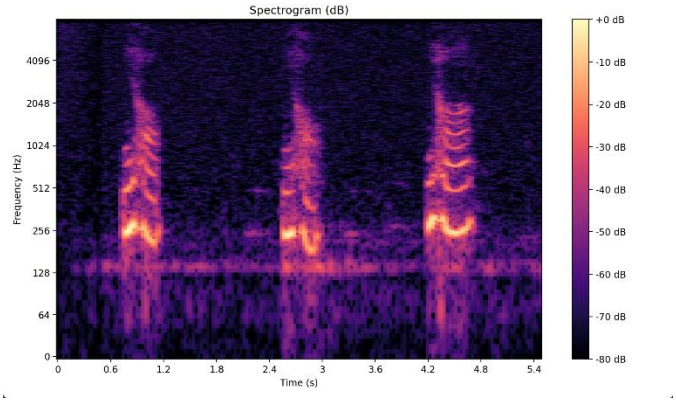

图 11 WebRTC 降噪频谱

观察 RNN 降噪方案的频谱，我们发现在第一次出现人声前并没有较好的滤波效果，但出现人声后，滤波效果极佳，这是由于 RNN 在处理时序数据时，其性能依赖于历史信息的积累和上下文的理解。在降噪任务中，RNN 需要从输入的时序音频数据中学习到背景噪声的特征以及人声的特征。然而，当音频刚开始时，RNN 并没有足够的上下文信息来准确区分噪声和人声。这是因为模型需要依赖记忆单元逐步积累相关的时序特征。

这种现象是 RNN 内部记忆机制的体现，也进一步说明了时序数据学习的渐进累积特性。

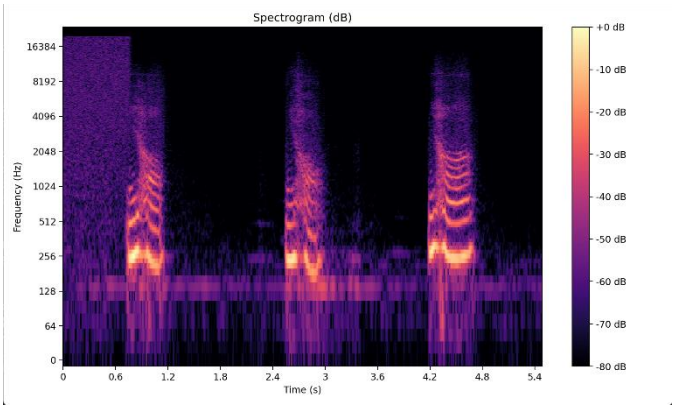

图 12 RNN 降噪频谱

表 1 给出了通过 speechmetric 计算的绝对音质指标 MOSNET [6] 和 SRMR[7][8][9]信息(均为越大越好)。

由于 RNN 最开始携带了部分白噪声，因而 MOSNET 的表现效果不如 WebRTC，但是在 SRMR 的对比上却出现了绝对的领先优势。

表格 1 白噪声降噪效果对比

	clean	noise	FIR	WebRTC	RNN
MOSNET	2.99	2.11	2.64	2.62	2.53
SRMR	17.64	16.57	17.99	21.74	22.74

3.2.2 模拟办公环境降噪性能

观察图 13 WebRTC 的降噪效果我们不难发现该算法将风扇的基频和谐波误判为人声信号进行了错误的加强，虽然在整个频域上仍然增加了信噪比，但是实际听感并不好，降噪效果虽有但不佳。

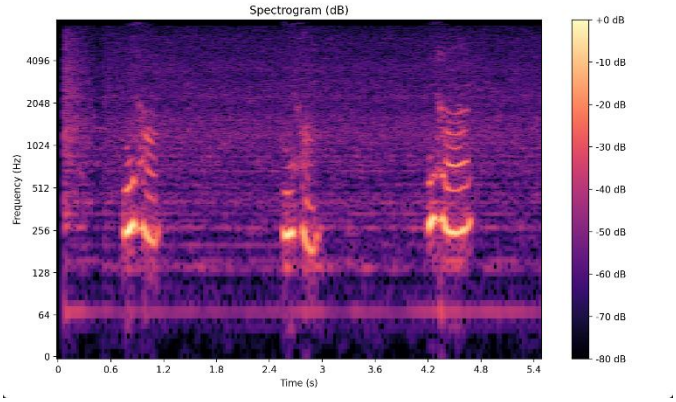

图 13 模拟环境下 WebRTC 降噪频谱

观察图 14 我们发现与白噪声的情况类似，RNN 神经网络模型在人声出现后很快便达到了最佳的降噪效果。由于神经网络存在人声存在或不存在的检测，我们发现无人声时的噪声抑制效果更佳。虽然人声存在时的噪声没有被完全的滤除，实际音频听感仍然存在噪音，但是其效果明显好于 WebRTC 的降噪算法。

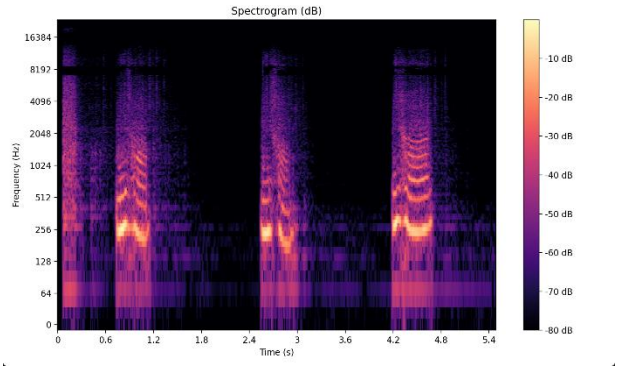

图 14 模拟环境下 RNN 降噪频谱

表 2 给出了模拟降噪环境的相关指标结果，可以很明显的看出 SRMR 参数的差别，WebRTC 降噪和 RNN 降噪均远远领先于未降噪和使用 FIR 的简单滤波效果，RNN 的降噪效果相较于 WebRTC 有更明显的优势。

表格 2 模拟环境降噪效果对比

	clean	noise	FIR	WebRTC	RNN
MOSNET	2.97	2.33	2.52	2.31	2.72
SRMR	17.66	6.15	8.19	19.04	19.51

4 总结

本文对基于物联网平台广播系统的语音降噪算法进行了研究和对比分析，主要聚焦于传统的 WebRTC 降噪算法和基于 RNN 的深度学习降噪算法的优劣。研究从理论分析、算法实现到实际性能评估等多个角度进行了系统性的探讨，得出了有意义的结论，并为实际应用场景提供了指导性建议。

首先，从理论层面来看，WebRTC 降噪算法基于经典的维纳滤波器理论，通过信号的频谱估计和动态调整实现降噪。其核心思想是通过分析语音信号与噪声的频谱特性，动态生成滤波器以最大化语音信号的保真度，同时最小化噪声对音质的干扰。这种传统的 DSP（数字信号处理）算法具有计算复杂度低、实时性强的特点，因此在语音通话等对实时性要求较高的应用场景中表现尤为出色。然而，在广播系统的应用场景中，WebRTC 降噪算法的局限性也逐渐显现。由于该算法主要依赖于信号的短时特征分析，对复杂背景噪声的处理能力有限，尤其是在混合多频噪声或类似音乐信号的非语音噪声环境下，可能会出现降噪不足或者信号失真的问题。此外，WebRTC 中的维纳滤波器对语音增强的支持有限，在噪声与语音信号频谱重叠严重时，其降噪性能会显著下降，甚至会误判部分噪声为人声信号，从而导致噪声被错误放大。

相比之下，基于 RNN 的降噪算法通过深度学习的方式克服了传统 DSP 算法的局限性。这种方法结合了 DSP 和神经网络的优势，将语音降噪任务分解为若干个子任务，如音高分析、特征提取、语音活动检测、噪声特征提取等，并在此基础上引入了 RNN（特别是 GRU 架构）来处理时序数据。通过对输入信号时序特征的学习和建模，该算法能够更好地识别语音与噪声之间的差异，尤其是在复杂噪声环境中表现出更强的适应性。RNN 利用其独特的记忆机制，通过积累历史信息来增强对信号的理解，从而在降噪效果和音质保真度之间达到了更好的平衡。例如，在本文的实验中，RNN 降噪算法在人声出现后能够迅速调整降噪策略，从而实现对语音信号的精确增强和对背景噪声的有效抑制。此外，该算法在无语音信号的区间内也表现出了较高的噪声抑制能力，使得广播系统在静音状态下的背景噪声显著降低，提升了整体的用户体验。

从实验结果来看，WebRTC 和 RNN 两种降噪算法在不同的噪声环境下表现出了一定的差异。在白噪声降噪实验中，WebRTC 的降噪效果较为稳定，其对基频和谐波的增强效果较好，但在高频段的噪声抑制能力有所不足，导致听感上仍存一定的噪声残留。而 RNN 降噪算法虽然在开始时由于上下文信息不足表现稍逊，但在人声出现后迅速达到最佳降噪效果，并且在整体音质指标（如 SRMR）上明显优于 WebRTC。这说明 RNN 能够更有效地处理复杂的噪声信号，尤其是在噪声频谱较为复杂的情况下。对于模拟办公环境的降噪实验，WebRTC 由于误判风扇噪声的部分频率为人声信号，从而对噪声进行了错误的增强，表现出降噪能力不足的问题。而 RNN 算法则充分利用了神经网络的特征提取能力，对风扇噪声进行了更准确的识别和抑制，同时在人声增强方面也表现出更高的细腻度，从而在主观听感和客观指标上均优于 WebRTC。

此外，RNN 的另一个优势在于其灵活性和可扩展性。通过调整神经网络的架构与训练数据集，RNN 降噪算法可以在更广泛的场景中实现定制化的降噪效果。例如，在训练过程中融入针对特定场景的噪声数据（如电风扇、电磁干扰、室外环境噪声等），可以进一步提升算法的针对性和鲁棒性。而且，RNN 与传统 DSP 算法的混合使用也为未来的语音降噪算法设计提供了新的方向。通过将 RNN 用于复杂噪声的处理，同时利用 DSP 算法处理简单噪声或高实时性任务，可以充分发挥两者的优势，实现更高效的降噪效果。

本文通过理论分析、实验验证和性能对比，全面评估了 WebRTC 降噪算法和基于 RNN 的深度学习降噪算法在物联网广播系统中的应用效果。研究表明，RNN 降噪算法在复杂噪声环境和音质保真度方面相较于 WebRTC 具有显著优势，尤其适用于对音质要求较高且实时性要求相对宽松的广播场景。随着物联网嵌

入设备的边缘计算能力性能加强，RNN 算法支持的降噪应用将有更大的用武之地。

参考文献:

[1] Chromium Project. WebRTC: Noise Suppression Module Source Code [EB/OL]. Available: https://chromium.googlesource.com/external/webrtc/+/ad34dbe934/webrtc/modules/audio_processing/ns, [Accessed: Oct. 2023].

[2] Valin, J.-M., "A Hybrid DSP/Deep Learning Approach to Real-Time Full-Band Speech Enhancement," *Mozilla Corporation*, Mountain View, CA, USA, 2017. [Online]. Available: AHybridDSP/DeepLearningApproachtoReal-TimeFull-BandSpeechEnhancement.

[3] Zhihan Gao, "RNNoise: Learning Noise Suppression," GitHub Repository, 2017. [Online]. Available:https://github.com/cpuimage/rnnoise

[4] Google, "WebRTC Noise Suppression (WebRTC_NS)," GitHub Repository, 2018. [Online]. Available: https://github.com/cpuimage/WebRTC_NS

[5] A. Liutkus, "Speechmetrics: A Python Package for Speech Quality and Intelligibility Metrics," GitHub Repository, 2020. [Online]. Available: https://github.com/aliutkus/speechmetrics.

[6] C.-C. Lo, S.-W. Fu, W.-C. Huang, X. Wang, J. Yamagishi, Y. Tsao, and H.-M. Wang, "MOSNet: Deep Learning Based Objective Assessment for Voice Conversion," *arXiv preprint arXiv:1904.08352*, Apr. 2019. [Online]. Available: https://arxiv.org/abs/1904.08352

[7] T. H. Falk, C. Zheng, and W.-Y. Chan, "A Non-Intrusive Quality and Intelligibility Measure of Reverberant and Dereverberated Speech," *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 7, pp. 1766–1774, Sept. 2010, doi: 10.1109/TASL.2010.2052247.

[8] J. F. Santos, M. Senoussaoui, and T. H. Falk, "An Updated Objective Intelligibility Estimation Metric for Normal Hearing Listeners Under Noise and Reverberation," in *Proc. International Workshop on Acoustic Signal Enhancement (IWAENC)*, Sept. 2014.

[9] J. F. Santos and T. H. Falk, "Updating the SRMR Metric for Improved Intelligibility Prediction for Cochlear Implant Users," *IEEE Trans. Audio Speech Lang. Process.*, Dec. 2014, doi: 10.1109/TASLP.2014.2363788.