软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(5):1569−1586 [doi: 10.13328/j.cnki.jos.006549]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

针对黑盒智能语音软件的对抗样本生成方法*

袁天昊[1,2], 吉顺慧[1,2], 张鹏程[1,2], 蔡涵博[1,2], 戴启印[1,2], 叶仕俊[1,2], 任 彬[1,2]

[1](水利部水利大数据重点实验室 (河海大学), 江苏 南京 211100)

[2](河海大学 计算机与信息学院, 江苏 南京 211100)

通信作者: 吉顺慧, E-mail: shunhuiji@hhu.edu.cn

摘 要: 随着深度学习技术的成熟, 智能语音识别软件获得了广泛的应用, 存在于智能软件内部的各种深度神经网络发挥了关键性的作用. 然而, 最近的研究表明: 含有微小扰动的对抗样本会对深度神经网络的安全性和鲁棒性构成极大威胁. 研究人员通常将生成的对抗样本作为测试用例输入到智能语音识别软件中, 观察对抗样本是否会让软件产生错误判断, 从而采取防御方法来提高智能软件安全性和鲁棒性. 在对抗样本的生成中, 黑盒智能语音软件在生活中较为常见, 具有实际的研究价值, 而现有的生成方法却存在一定的局限性. 为此, 针对黑盒智能语音软件, 提出了一种基于萤火虫算法和梯度评估方法的目标对抗样本生成方法, 即萤火虫-梯度对抗样本生成方法. 针对设定的目标文本, 在原始的音频样本中不断加入干扰, 根据当前对抗样本的文本内容与目标文本之间的编辑距离, 选择使用萤火虫算法或梯度评估方法来优化对抗样本, 最终生成目标对抗样本. 为了验证方法的效果, 在常用的语音识别软件上, 使用公共语音数据集、谷歌命令数据集和 LibriSpeech 数据集这 3 种不同类型的语音数据集进行了实验评估, 并寻找志愿者进行对抗样本的质量评估. 实验表明, 提出的方法能有效提高目标对抗样本生成的成功率, 例如针对 DeepSpeech 语音识别软件, 在公共语音数据集上生成对抗样本的成功率相比对比方法提升了 13%.

关键词: 智能软件; 语音识别; 对抗样本; 萤火虫算法; 梯度评估方法

中图法分类号: TP311

中文引用格式: 袁天昊, 吉顺慧, 张鹏程, 蔡涵博, 戴启印, 叶仕俊, 任彬. 针对黑盒智能语音软件的对抗样本生成方法. 软件学报, 2022, 33(5): 1569–1586. http://www.jos.org.cn/1000-9825/6549.htm

英文引用格式: Yuan TH, Ji SH, Zhang PC, Cai HB, Dai QY, Ye SJ, Ren B. Adversarial Example Generation Method for Black Box Intelligent Speech Software. Ruan Jian Xue Bao/Journal of Software, 2022, 33(5): 1569–1586 (in Chinese). http://www.jos.org.cn/1000-9825/6549.htm

Adversarial Example Generation Method for Black Box Intelligent Speech Software

YUAN Tian-Hao[1,2], JI Shun-Hui[1,2], ZHANG Peng-Cheng[1,2], CAI Han-Bo[1,2], DAI Qi-Yin[1,2], YE Shi-Jun[1,2], REN Bin[1,2]

[1](Key Laboratory of Water Big Data Technology of Ministry of Water Resources (Hohai University), Nanjing 211100, China)

[2](College of Computer and Information, Hohai University, Nanjing 211100, China)

Abstract: With the maturity of deep learning technology, intelligent speech recognition software has been widely used. Various deep neural networks in the intelligent software play a crucial role. Recent studies have shown that minor disturbances in adversarial examples significantly threaten the security and robustness of deep neural networks. Researchers usually take the generated adversarial examples as the test cases and input them into the intelligent speech recognition software to test whether the adversarial examples will make the software misjudge. And then defense methods are adopted to improve the security and robustness of intelligent software. For the adversarial example generation, black box intelligent speech software is more common in life and has practical research value. However, the existing generation methods have some limitations. Therefore, this study proposes a target adversarial example generation method for

* 基金项目: 国家自然科学基金 (U21B2016, 61702159); 江苏省自然科学基金 (BK20191297); 中央高校基本科研业务费
本文由“领域软件工程”专题特约编辑汤恩义副教授、江贺教授、陈俊洁副教授、李必信教授以及唐滨副教授推荐.
收稿时间: 2021-08-08; 修改时间: 2021-10-09; 采用时间: 2022-01-10; jos 在线出版时间: 2022-01-28

the black box speech software based on the firefly algorithm and gradient evaluation method, namely the firefly-gradient adversarial example generation method. With the set target text, disturbances are added to the original speech example. The firefly algorithm or gradient evaluation method is chosen to optimize the adversarial example according to the edit distance between the text of the current generated adversarial example and the target text so that the target adversarial example is generated finally. To verify the effectiveness of the method, this study conducts an experimental evaluation on common speech recognition software, using three different types of speech datasets: Common Speech dataset, Google Command dataset and LibriSpeech dataset, and looks for volunteers to evaluate the generated adversarial examples. Experimental results show that the proposed method can effectively improve the success rate of target adversarial example generation. For example, for the DeepSpeech speech recognition software, the success rate of generating adversarial examples on Common Speech datasets is 13% higher than that of the compared method.

Key words: intelligent software; speech recognition; adversarial examples; firefly algorithm; gradient evaluation method

近年来, 随着机器学习, 尤其是深度学习技术的成熟, 智能软件因其高效率、高准确率以及所需时间成本较低等优势, 受到人们欢迎并被广泛使用. 其中, 智能语音软件, 例如科大讯飞的语音翻译、谷歌公司的语音助手、苹果手机的 Siri 等, 在生活中应用非常广泛. 此外, 智能语音识别软件在智能家居[1]、自动驾驶[2]等领域也取得了一定的进展. 智能语音识别软件之所以应用如此广泛, 存在于其内部的各种深度神经网络发挥了关键性的作用. 随着研究的不断深入, 人们对于含有深度神经网络模型的智能软件 (以下简称软件) 的安全性和鲁棒性等属性日趋重视.

然而, Goodfellow 等人[3]和 Szegedy 等人[4]的研究表明: 含有微小扰动的样本会对深度神经网络的安全性和鲁棒性构成极大威胁. 通过在原始样本中添加一些细微的非随机化扰动, 来生成人们察觉不出的异常样本, 然后将其输入到智能软件中. 面对这些异常样本, 软件往往会产生错误的判断, 进而暴露了相关软件安全性和鲁棒性的漏洞. 通过添加微小扰动而生成的异常样本, 被称作对抗样本. 为了解软件自身的潜在漏洞, 研究人员通常会使用各种方法生成对抗样本这种测试用例, 并输入到相关软件中, 来观察样本是否会让软件产生错误判断. 进而针对软件测试中暴露出的漏洞, 采取相应的防御方法, 提高软件的安全性和鲁棒性. 在智能语音识别软件领域, 对抗样本这种测试用例也被广泛应用[5]. 具体而言, 研究人员在输入的原始音频样本中加入一些细微的干扰噪声, 在保持对抗样本与原始样本音频内容一致的情况下, 使软件在判断的过程中发生错误, 最终生成与原始语音样本不相符的文本内容.

在语音对抗样本的研究中, 根据智能语音识别软件内部结构是否被人了解, 可以将对抗样本生成方法分为针对白盒软件和黑盒软件两种类型. 白盒软件是指人们已经了解其内部结构和参数的语音识别软件, 根据这些信息采取相应的方法来生成对抗样本. 反之, 对于人们不了解其内部结构和参数, 只能从输入和输出上进行观察的智能语音识别软件, 则称为黑盒软件. 在实际应用中, 多数智能语音识别软件是黑盒的, 例如谷歌公司的语音助手、苹果手机的 Siri, 普通用户无法了解这些软件的内部结构和参数. 另一方面, 根据样本是否指定生成目标, 又可以将对抗样本分为非目标对抗样本和目标对抗样本. 非目标对抗样本要求生成的对抗样本相应文本内容与原始样本不相同即可. 而目标对抗样本不仅要求对抗样本文本内容和原始样本不相同, 而且要求其文本内容与预先设置的目标文本完全相同. 在实际应用中, 非目标对抗样本会使被测软件做出任意错误的预测, 而目标对抗样本则会引导被测软件产生期望的错误, 后者通常会产生更具破坏性的影响. 例如, 在声控驾驶的场景中, 智能软件应该将语音命令识别为“turn left”. 对于非目标对抗样本, 软件可能会错误地将其识别为无效命令“turn loft”. 而在目标对抗样本中, 软件可能会将其识别为完全相反的命令“turn right”, 使得汽车转向改变, 甚至可能导致交通事故的发生. 因此, 针对黑盒智能语音软件的目标对抗样本生成具有实际价值, 是值得研究的.

针对黑盒语音识别系统已提出了一些对抗样本生成方法[6−8]. 但是, 这些方法仍然存在一些局限性.

(1) 在对黑盒智能语音软件进行对抗样本的生成时, 多数方法侧重于非目标对抗样本的生成, 而生成目标对抗样本作为测试用例的研究却不多.

(2) 一些方法在实验评估时采用的语音数据集较为单一, 只是运用一些包含简单单词、短语的语音数据集, 却很少使用包含中等句或长句内容的语音数据集, 这导致了这些对抗样本生成方法普适性不强, 往往只是针对某种特殊的语音数据集.

(3) 现有的针对黑盒智能语音软件的目标对抗样本生成方法, 成功率比较低. 例如 Taori 等人[6]在公共语音数据集上, 生成目标对抗样本的成功率只有 35%, 还有很大的提升空间.

由于萤火虫算法[9]的搜索范围比较广, 搜索跨度不会过大, 且梯度评估方法[6]能够用于对抗样本的局部优化. 因此, 本文提出了一种针对黑盒智能语音软件的目标对抗样本生成方法, 即萤火虫-梯度对抗样本生成方法, 用来解决现有方法存在的局限性. 对比现有的工作, 本文的贡献点包括以下 3 方面.

(1) 提出了一种基于萤火虫算法[9]与梯度评估方法[6]的目标对抗样本生成方法. 方法根据生成的对抗样本文本内容与设置的目标文本之间的编辑距离, 使用萤火虫算法或梯度评估方法来优化对抗样本, 不断进行迭代, 将原始的语音样本最终生成为目标对抗样本.

(2) 将提出的语音对抗样本生成方法应用在多个常用的语音数据集上, 进行实验评估. 这些语音数据集包括公共语音数据集[10]、谷歌命令语音数据集[11]、LibriSpeech 数据集[12]. 其中, 公共语音和 LibriSpeech 两种数据集包含中长句音频文件. 结果表明提出的方法适用于不同类型的语音数据集.

(3) 进行了大规模的实验来评估方法的效果, 并寻找志愿者测试生成的对抗样本对人听觉的影响. 结果表明, 与现有生成方法相比, 提出的方法针对不同的语音数据集, 对抗样本语音相似度和生成样本所需时间略差于对比方法的情况下, 生成对抗样本的成功率均优于其他方法. 例如在公共语音数据集上, 生成目标对抗样本的成功率相比基于遗传算法和梯度评估的对抗样本生成方法[6]提升了 13%.

本文第 1 节对所作研究的背景知识, 包括语音识别、萤火虫算法、梯度评估方法以及编辑距离进行基本介绍. 第 2 节对现有针对智能语音软件的对抗样本生成方法进行总结. 第 3 节对提出的萤火虫-梯度对抗样本生成方法进行详细介绍. 第 4 节介绍评估实验的设置, 包括使用的语音数据集、语音识别软件、方法衡量指标、对照方法以及参数选择. 第 5 节是对于实验结果的详细分析, 包含对生成对抗样本的人工验证. 第 6 节则对全文进行总结, 方法存在的问题以及未来的工作.

1 背 景

1.1 语音识别

语音识别是语音领域中的常见技术, 它的功能是通过智能语音识别软件理解输入的音频, 并将其正确转换成相应的文本内容. 传统语音识别软件[13−15]建立在似然概率的基础上, 原理比较复杂. 而目前常用的软件大多是端到端的智能语音识别软件[16], 可以将输入的声学特征序列映射到输出的词序列上. 如图 1 所示, 输入原始音频, 进行端到端智能语音识别系统的一系列步骤处理, 最终可以得到相应的文本内容. 一般来说, 端到端的智能语音识别处理主要包括 3 个步骤.

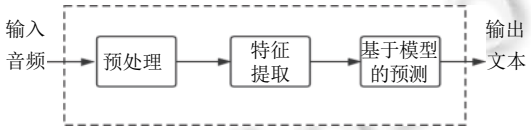

图 1 端到端语音识别系统

(1) 预处理: 对信号量低于阈值的时间域进行删除, 最常用的技术是语音活动检测技术 (VAD)[17].

(2) 特征提取: 就是从语音信号中提取出相关的语音特征. 常用的语音特征提取方法有梅尔倒谱系数 (MFCC)[18], 线性预测系数 (LPC)[19], 感知线性预测 (PLP)[20]. 其中, MFCC 是最常用的一种特征提取方法. 它的主要原理就是通过反向傅里叶变换, 将音频先转换到频域, 然后再将其转换到时间域.

(3) 基于模型的预测: 将提取的特征作为输入, 和软件模型进行匹配来得到预测的结果. 当前的软件模型大多采用具有联结主义时间分类 (connectionist temporal classification, CTC) 损失函数[21]的循环神经网络 (RNN), 它的优点在于较为简单, 且方便实用. CTC 损失函数是语音领域中常用的一种损失函数. 相较于传统的语音识别软件要求生成的文本和语音保持对齐, CTC 损失函数不需要二者的严格对齐, 处理起来更为方便.

1.2 萤火虫算法

萤火虫算法属于群智能算法[22], 是研究人员根据自然界夜晚萤火虫的聚集行为而提出的一种启发式算法. 在夏天的晚上, 萤火虫自身会发出短暂而有节奏的独特亮光, 它们具有吸引配偶、繁衍后代和寻找猎物、进行捕食的作用. 一般来说, 在发光的种群中, 每一只萤火虫总是朝着比自己亮度更高的萤火虫所在位置进行移动. 同时, 亮度不同的萤火虫之间还存在一定的吸引度. 在进行多次移动之后, 其他萤火虫会聚集到亮度最高的萤火虫所在位置周围. 为了对问题进行简化, Yang 等人[23]对萤火虫算法的具体细节进行了 3 种处理.

(1) 萤火虫之间的吸引不考虑性别, 即不考虑这种吸引属于同性吸引还是异性吸引.

(2) 萤火虫之间的吸引度与它们的亮度相关联. 因此, 对于任意两只萤火虫来说, 亮度低的萤火虫将会朝着光度高的物种所在位置进行移动. 同时, 二者之间的吸引度和亮度都随着二者之间的距离增大而减小, 反之会增加吸引度. 对于萤火虫来说, 如果没有任何其他萤火虫的亮度比它的高度高, 那么它就会在原位置周围进行随机移动.

(3) 萤火虫亮度 I 的数值取决于实验需要优化的目标函数的数值.

在萤火虫算法中, 有两个重要的部分: 亮度 I 的变化和吸引度 β 的表达形式. 前者的数值由实验使用的目标函数来决定, 而后者则根据相邻两只萤火虫之间的距离而变化. 吸引度 β 的表达式为:

$$\beta = \beta_0 \mathrm{e}^{-\gamma r^2} \tag{1}$$

其中, β_0 表示两只萤火虫距离 r 为 0 时的吸引度, 其中 r 表示两只萤火虫之间的距离, γ 表示光吸收系数, 通常取值为 1. 例如, 两只萤火虫 i, j 在位置 x_i, x_j 之间的距离可以表示为:

$$r_{ij} = \left\| x_i - x_j \right\|_2 \tag{2}$$

其中, "$\| \|_2$"这种格式是范式的表现形式.

假设一只萤火虫 j 的亮度比另一只萤火虫 i 高, 那么 i 将会被 j 所吸引, 从而进行位置的移动. 萤火虫 i 位置移动的公式为:

$$x_i = x_i + \beta_0 \mathrm{e}^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha \varepsilon_i \tag{3}$$

其中, 公式右侧第 1 项表示萤火虫 i 原来所在的位置, 第 2 项表示两只萤火虫之间的吸引度, 第 3 项表示随机移动, ε_i 表示一种数学分布, 例如 0-1 均匀分布、高斯分布等. α 表示分布的系数, 通常的取值为 0 到 1 之间的常数[24].

1.3 梯度评估

梯度评估方法, 是一种针对黑盒智能软件的对抗样本生成方法. 受白盒生成方法的启发, 其直接对黑盒软件内部的梯度进行评估, 以支持黑盒软件的对抗样本生成, 提出了一种基于有限差分法[25]的梯度评估对抗样本生成方法. 后来又提出了基于主成分分析[26]的梯度评估对抗样本生成方法, Taori 等人[6]将这种方法用于智能语音软件的对抗样本生成, 并取得了不错的效果.

本文采用基于有限差分法的梯度评估对抗样本生成方法. 假设软件的梯度函数为$g(X)$, 输入的函数是一个维度为 d 维的向量 X, 其中向量中每一个元素表示为 Xi, i 的取值范围 $[1, d]$, 正则基向量表示为 e_i, 表示只有在第 i 个位置上数字为 1, 其余的位置数字都为 0. 梯度评估的相关公式为:

$$FD_x(g(x), \delta) = \begin{bmatrix} (g(x+\delta e_1) - g(x))/\delta \\ \cdots \\ (g(x+\delta ed) - g(x))/\delta \end{bmatrix} \tag{4}$$

其中, δ 表示一个无限接近于 0 的极小值. 该公式的主要作用就是生成梯度, 将其与原始的梯度函数值做差值, 对原始的梯度函数值做微小的改动, 进行优化, 从而生成更精准的对抗样本. 该公式借鉴了数学上有关差分的思想, 因此被称为基于有限差分法的梯度评估方法.

1.4 编辑距离

编辑距离[27]是一种用来衡量两种序列之间差异的指标, 是指其中任意一个序列转换成另一个序列所需要的单个字符编辑操作的最小次数. 常用的单字符操作有 3 种方式: 插入、删除以及替换. 设置两个序列 w_1 与 w_2, 序列内容分别为"left"和"right", 从"left"转换成"right", 无论是将字符"l""e""f"分别替换成"r""i""g", 再插入一个字符

“h”, 还是将字符“l”“e”“f”分别替换成“i”“g”“h”, 再插入一个字符“r”, 所需要的操作步骤都为 4 步. 因此, “left”转换成“right”所需的编辑距离为 4.

2 相关工作

目前对于图像领域对抗样本生成方法的研究[3,28−30]比较多, 而对于智能语音识别软件对抗样本生成方法的研究则相对较少, 这是由智能语音软件自身特点所决定的. 一方面, 智能语音识别软件系统远比图像分类软件系统复杂, 因为前者有一个时间维度需要处理, 后者则不存在这样的问题. 另一方面, 音频文件的采样率通常较高. 例如, 采样率为 15 kHz 的音频样本, 意味着处理该样本需要每秒采样 15 000 次, 处理的数据量远大于图像. 因此, 语音识别对抗样本的研究比图像对抗样本研究更为复杂. 然而, 随着智能语音软件在生产和生活中的广泛应用, 对于该种软件的安全性与鲁棒性, 以及对抗样本的相关研究逐步为人们所重视.

Carlini 等人[31]针对白盒智能语音软件, 提出了一种通过 MFC 层来传递梯度的方法, 通过梯度来连接到原始的语音输入, 并将这种方法运用在选择的测试语音软件上, 取得了成功. Cisse 等人[32]将图像领域的快速符号梯度方法 (FGSM) 应用到基于注意力机制的深度关键词识别软件上, 生成了对抗样本. Schonherr 等人[33]提出了一种基于心理学声学知识的对抗样本生成方法, 并且针对特定的语音软件进行对抗样本生成. 和 Schonherr 的研究类似, Qin 等人[34]也使用了心理学方面的知识, 将心理声学原理和相关的梯度技术相结合, 生成了对抗样本, 并使得生成的对抗样本更加具有隐蔽性和鲁棒性.

以上所提出的都是针对白盒智能语音识别软件的对抗样本生成方法, 是在用于了解语音识别软件内部参数的情况下而采用的方法. 然而, 对于内部参数和结构都无法获得的黑盒智能语音软件, 上述方法就会失去作用. 因此, 研究人员对于黑盒语音识别软件的对抗样本生成也进行了研究.

Alzantot 等人[35]采用标准的遗传算法, 生成了针对黑盒智能语音软件的对抗样本, 并在谷歌命令数据集上进行单个命令之间的转换, 取得了不错的效果. Taori 等人[6]在此研究的基础上, 使用优化过的、带有动态变异机制的遗传算法与梯度评估相融合的方法, 将音频转化成指定的目标短语, 并任意选取了公共语音数据集上的 100 个公共语音音频样本进行实验, 生成目标对抗样本的成功率为 35%. Du 等人[36]使用了粒子群优化算法和欺骗梯度算法融合的算法来生成对抗样本, 并选取了多种智能语音软件作为测试对象, 包括黑盒和白盒, 语音分类和识别等多种软件, 都获得了成功. Zhang 等人[37]利用麦克风自身存在的缺陷, 通过调制超声波的方法, 提出了海豚攻击, 这种攻击是人耳听不见到的攻击, 并且在流行的一些语音识别软件上进行了语音唤醒和语音识别的实验. Chen 等人[7]采用了布谷鸟搜索算法来生成面向黑盒软件的语音对抗样本. Khare 等人[8]提出了一个基于优化算法的多目标优化的框架, 通过这个框架来生成语音对抗样本, 并对 DeepSpeech[38]和 Kaldi[39]两种常用的智能语音识别软件, 进行了有目标的和无目标的攻击. Gong 等人[40]采用了强化学习的思想来生成对抗样本, 具有实时性. Esmaeilpour 等人[41]通过减小噪声干扰的方法来生成对抗样本, 并在 DeepSpeech、Kaldi 等智能语音软件上进行了实验.

3 基于萤火虫算法与梯度评估方法的对抗样本生成方法

3.1 方法总体框架

本文设计的萤火虫-梯度对抗样本生成方法, 针对内部结构和参数未知的黑盒语音软件, 通过萤火虫算法以及梯度评估方法这两种优异的优化算法, 向原始音频样本中加入干扰噪声, 不断进行迭代, 最终生成目标对抗样本. 方法的系统框架如图 2 所示, 主要分为 3 个部分.

(1) 种群初始化: 以萤火虫算法为代表的群智能算法的思想, 是在含有多个个体的种群中进行优化操作, 寻找满足条件的最优个体. 同样, 梯度评估方法也需要对多个个体进行梯度处理, 最终挑选出最优个体. 然而, 原始音频样本仅提供了一个样本个体, 不满足萤火虫算法和梯度评估方法的使用条件. 因此, 有必要对原始样本进行扩充, 形成包含多个样本个体的种群. 另外, 为了使个体间产生差异, 还需要向种群中加入随机噪声, 完成种群初始化操作.

(2) 初始对抗样本生成: 种群初始化完成后, 根据选择的适应度函数, 给种群中所有个体进行评分, 寻找到当前

满足条件的最优个体, 将其作为初始的对抗样本, 并输入到语音软件中, 得到样本对应的文本内容. 为了解生成对抗样本文本内容与目标文本的差距, 采用编辑距离这一指标来衡量对抗样本的文本内容转换到目标文本所需操作次数.

(3) 目标对抗样本生成: 生成初始对抗样本后, 使用萤火虫算法或梯度评估方法继续进行优化, 样本优化算法的选择取决于对抗样本的文本内容与目标文本之间的编辑距离: 当其小于一定的数值时, 采用萤火虫算法进行寻优的效果将会减弱, 则采用梯度评估方法进行对抗样本的局部优化. 经过不断迭代, 如果生成对抗样本的文本内容与目标文本之间的编辑距离为 0, 意味着样本对应的文本内容和设置的目标文本完全相同, 则成功地生成了目标对抗样本.

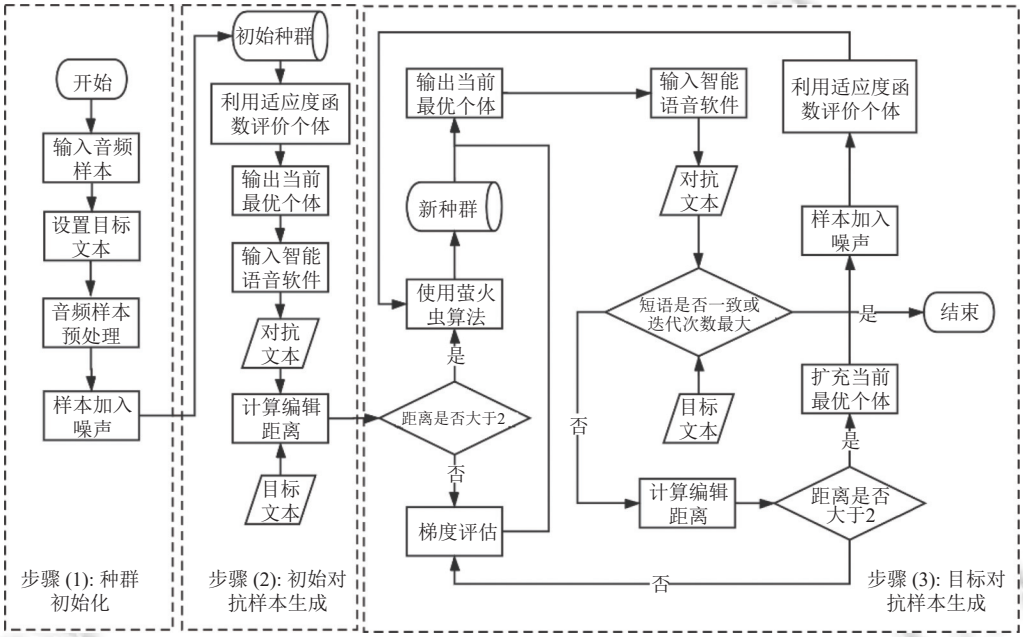

图 2 方法总体框架

3.2 种群初始化

给定原始音频样本 x, 设置目标文本 t, 最大迭代次数 $epochMax$, 萤火虫个体数量 n 等数值, 期望生成一个目标对抗样本, 即 x_{adv}. 例如给定原始内容为“hello world”的音频样本, 设置目标文本为“for help”, 使用萤火虫-梯度对抗样本生成方法, 基于萤火虫算法和梯度评估方法, 期望生成一个被智能语音软件识别成“for help”的目标对抗样本.

首先, 需要构造出包含多个样本个体的种群, 以便于在后续工作中使用萤火虫算法和梯度评估方法, 在种群中寻找最优样本个体. 然而, 目前的输入只有一个原始音频样本, 不存在包含多个样本的种群. 因此, 需要对输入个体进行扩充操作: 将原始音频样本转换成一维矩阵, 并扩充成为 n 行相同的多维矩阵形式, 其中多维矩阵的每一行代表一个样本个体, 这样就形成了含有 n 个相同样本的种群.

由原始的音频样本扩充生成的样本种群, 个体之间不存在差异. 在这种情况下无法选择出最优个体. 为此, 可以通过向种群中所有个体随机添加细微噪声的方式, 使得个体间形成一定差异. 加入的细微噪声可以当做原始音频样本中的背景噪声. 为了防止噪声过大而对原始音频样本的内容造成干扰, 导致用户无法听清原始音频的内容, 需要对噪声进行特定处理. 根据 Reichenbach 等人[42]的研究: 相较于高频率的声音, 人耳对于低频率的声音更为敏感. 因此, 方法对于所加入的噪声使用截止频率为 7 kHz 的高通滤波器进行处理, 这样做的目的是保留人耳相对不敏感的高频率噪声, 让人认为在原始样本加入的噪声属于背景噪声而无需重视, 这种噪声处理是贯穿全过程的, 在第 3.4 节中还将使用到.

3.3 初始对抗样本生成

在优化方法中, 通常需要一种衡量种群个体表现的函数, 称为适应度函数. 优化方法会根据样本个体的适应度函数值来选择当前最优个体, 不断进行优化. 针对特定问题, 一般需要人为选择适当的函数, 来作为算法的适应度函数. 适应度函数的选择对于总体方法的性能表现来说是非常重要的.

在本方法中, 使用语音领域常用的 CTC 损失函数[21]作为总体方法的适应度函数. 选择该函数的原因主要有以下两点.

(1) CTC 损失函数在使用时不要求原始音频样本与对应文本内容间的严格对齐, 处理起来更为方便.

(2) CTC 损失函数可以有效衡量生成对抗样本的文本内容与目标文本之间的接近程度, 即损失函数的数值越小, 意味着二者的文本内容越接近.

因此, 对抗样本的生成需要不断减小损失函数, 即适应度函数值, 使得样本的文本内容与目标文本越来越接近. 使用适应度函数对初始化后的个体进行评分, 得到相应的数值, 并使用排序算法按适应度函数值从低到高对种群个体进行排序, 记录最低适应度函数值, 生成初始语音对抗样本. 将生成的对抗样本输入到选择的黑盒智能语音软件中, 得到相应的文本内容.

尽管适应度函数值能够衡量对抗样本的文本内容与目标文本之间的接近程度, 然而, 这对于用户来说并不容易理解: 用户只是知晓函数值在不断下降, 但是不清楚下降到什么数值时会生成最终的目标对抗样本. 并且用户也不了解每一轮迭代后生成对抗样本的文本内容与目标文本二者之间的具体差异, 即对抗样本的文本内容还需要多少次文本转换操作才能最终变成目标文本. 因此, 需要采用一种衡量文本之间转换所需操作次数的指标. 由于编辑距离在衡量字符串之间转换次数的优异表现, 因此, 方法使用编辑距离这一常用指标来衡量对抗样本文本内容与目标文本之间的具体差异.

3.4 目标对抗样本生成

大量实验表明, 萤火虫算法的优化效果优于遗传算法和粒子群优化算法. 而梯度评估方法对于样本局部区域的优化效果也比较好. 因此, 根据编辑距离的不同情况, 选择使用萤火虫算法或梯度评估方法, 来对生成的对抗样本进行优化操作. 下面将对这两种算法的使用进行详细介绍.

(1) 萤火虫算法: 目前为止, 只是生成了初始的对抗样本. 为了生成目标对抗样本, 还需使用萤火虫算法不断进行迭代, 来降低最优个体的适应度函数值, 以及对抗样本文本内容与目标文本之间的编辑距离 (以下简称编辑距离). 为了继续寻找全局最优个体以及对应的适应度函数值, 还需要继续对当前最优个体的矩阵形式进行扩充, 形成行数为 n 的多维矩阵, 即使用本轮迭代生成的最优个体, 再次初始化种群. 然后将种群中的所有个体加入随机噪声, 来扩大种群个体间适应度函数值的差异, 形成新的种群. 和种群初始化噪声的处理类似, 在新种群中所加入的随机噪声也是经过高通滤波器处理过的高频率噪声.

形成新种群后, 采用萤火虫算法进行种群个体间的位置移动. 设定种群中萤火虫个体的适应度函数值和所在位置有关, 而个体位置可以用个体代表的一维矩阵的数值来表示. 如果种群中任意两个个体之间的适应度函数值不同, 那么适应度值高的个体会在相应一维矩阵的所有维度上向函数值低的个体所在位置进行移动, 即一维矩阵的数值发生改变, 而种群中适应度函数值最低的个体则会在原位置周围进行随机移动, 即一维矩阵的数值发生微小的改变. 其余的萤火虫个体会向最优个体所在位置进行靠拢. 在萤火虫个体移动的同时, 任意两个适应度不同的个体间的吸引度也会随着距离远近发生改变. 等待种群中全部个体的移动结束之后, 种群个体所在位置会发生一定的改变, 相应的适应度函数值. 经过萤火虫算法的不断迭代后, 最优位置将会被找到. 在个体随机移动方法的选择上, 本方法采用–1 到 1 的均匀分布作为个体位置的随机移动方式. 根据适应度函数值, 使用排序算法对所有个体进行排序并记录下当前适应度函数最优个体并记录最优函数值, 生成相应的对抗样本. 将当前最优个体输入到黑盒智能语音软件中, 得到对抗样本的文本内容, 计算新的文本内容与目标文本之间的编辑距离. 将新的最优个体的矩阵形式进行扩充, 不断重复上面的操作. 有关萤火虫算法的详细步骤如算法 1 所示.

在经过萤火虫算法的多次迭代后, 种群最优个体适应度函数值以及编辑距离都会不断减小, 使得生成对抗样本的文本内容越来越接近目标文本. 例如, 将原始内容为“hello world”的音频样本加入扰动, 逐渐生成诸如“hell

word”“fel horp”“for hp”等文本内容. 可以观察到, 随着算法的不断迭代, 生成对抗样本的文本内容向目标文本“for help”不断靠近.

算法 1. 萤火虫算法.

输入: 原始语音样本 x, 设置的目标短语 t, 最大迭代次数 *epochMax*, 种群数量 n;
输出: 一个目标语音对抗样本 x_{adv}.

初始化 *iter* = 0 并设置 CTC 损失函数作为适应度函数
While *iter* <*epochMax* and *Decode*(最优种群)!=*t* **do**
scores ← –*CTCLoss*(*n*, *t*)
当前最优萤火虫个体 ← *population*[Argmax(*scores*)]
扩充当前最优萤火虫的个体数量, 形成萤火虫新种群
给新种群个体加入噪声
  for *i* =1:*n* //使用萤火虫算法
    for *j* = 1:*i*
      处于 X_i 位置的萤火虫亮度 I_i 由适应度函数 *f*(*x*) 决定
      if $I_j < I_i$
       将个体 *i* 在所有维度上朝着个体 *j* 所在位置移动
      end if
      个体之间吸引度随着距离而变化
      更新种群中个体的适应度函数值
      end for *j*
      end for *i*
    根据适应度函数值对种群个体进行排序, 找到当前最优个体
  return 最优萤火虫个体
end while

(2) 梯度评估方法: 萤火虫算法主要用于全局优化, 即在一个比较大的范围内寻找当前最优个体. 然而, 当生成对抗样本的文本内容十分接近目标文本, 即当前编辑距离小于一定的数值时, 如果继续使用萤火虫算法, 寻优效果将会大大减弱. 例如当前生成的对抗样本内容为“for hp”, 距离目标文本“for help”只有 2 个编辑距离时, 使用萤火虫算法, 不论如何加入噪声, 种群个体一维矩阵的数值只会发生微小改变, 即种群中样本个体的位置相对固定, 不会进行显著移动. 种群中寻找到的最优个体长期保持不变, 只是在原始位置周围进行非常微小的随机移动, 这样就会导致最优个体适应度函数值下降程度相比之前较为缓慢, 而编辑距离也会长时间保持不变, 直至达到最大的迭代次数, 生成对抗样本的过程就会陷入僵局. 由于此时生成对抗样本的文本内容已经非常接近目标文本, 只需要在对抗样本的关键区域加入一些细微扰动, 就可以最终生成目标对抗样本. 因此, 需要采用另外的方法对生成的对抗样本进行局部优化, 在不改变最优种群个体的情况下, 也能最终生成目标对抗样本.

为此, 我们采用基于有限差分法的梯度评估对抗样本生成方法, 把适应度函数值当做梯度评估方法中的梯度对象. 这样做的目的是在不改变种群个体的情况下, 对样本梯度进行处理, 减小梯度与编辑距离的值, 使其能够最终生成目标对抗样本. 首先, 需要将当前最优个体的矩阵形式进行扩充, 形成 n 行数值相同的多维矩阵形式. 与萤火虫算法的处理步骤不同的是, 此时不需要再加入噪声形成种群间的差异. 而是在原种群的基础上对每个种群个体进行微小的干扰, 然后使用适应度函数进行评估, 得到干扰后的种群个体的适应度函数值, 即 *f*(*x*) 的值. 使用梯度的相关公式求出相应的梯度值, 并在适应度函数值的基础上减去这个梯度值, 就会得到当前对抗样本的适应度函数值. 继续使用该算法进行梯度下降处理, 可以最终生成目标对抗样本. 梯度评估方法的详细步骤如算法 2.

在总体方法中, 根据 Taori 等人[6]的研究经验, 将使用梯度评估方法的编辑距离阈值设置为 2, 即编辑距离大于 2 时, 采用萤火虫算法进行优化; 反之, 则采用梯度评估方法进行局部的优化.

算法 2. 梯度评估方法.

输入: 原始语音样本 x, 设置的目标短语 t, 最大迭代次数 *epochMax*, 种群数量 n;
输出: 一个目标语音对抗样本 x_{adv}.

初始化 *iter* = 0 并设置 CTC 损失函数作为适应度函数
While *iter* <*maxIters* and *Decode*(最优种群)!=*t* **do**
scores ← –*CTCLoss*(*n*, *t*)
当前最优萤火虫个体 ← *population*[Argmax(*scores*)]
扩充当前最优萤火虫的个体数量, 形成新种群
给新种群中每个个体增加一些细微的干扰
更新种群中个体的适应度函数值 *f(x)*
使用公式 *grad*=(*f*(*x*)–*scores*)/变异率, 得到梯度值 *grad*
使用公式 *f*(*x*)= *f*(*x*)–*grad*
return *f(x)*
end while

提出的方法将萤火虫算法与梯度评估方法进行结合, 即使用萤火虫算法在全局上进行扰动, 使用梯度评估方法进行局部关键扰动, 不断进行迭代, 生成目标对抗样本. 当迭代次数达到设置的最大次数, 或生成对抗样本的文本内容与设置的目标文本完全一致的条件下, 方法运行结束. 如果最终生成的对抗文本和目标文本之间的编辑距离为 0 时, 表示使用萤火虫-梯度方法, 在原始的音频样本上成功生成了目标对抗样本.

4 实验设置

本节对实验设置进行介绍, 用于评估提出的萤火虫-梯度评估方法生成对抗样本的实际效果, 主要从生成目标对抗样本的语音相似度、生成对抗样本所需时间以及成功率这 3 种指标来进行实验评估.

4.1 实验数据集

实验选择公共语音数据集[10]上的任意 100 个音频样本, 谷歌命令语音数据集[11]上的任意 10 种类型的共 100 个语音命令样本以及 LibriSpeech 语音数据集[12]上的任意 50 个音频样本, 来进行方法的效果评估. 其中, 公共语音数据集是由 Mozilla 公司发布的大规模语音数据集, 其中的音频文件为中等长度的一句话, 例如 The Boy was surprised 等; 谷歌命令语音数据集是由谷歌公司发布的, 拥有多种常用单词命令、长度为 1 s 的发音, 例如 yes、no、left、right 等; LibriSpeech 语音数据集则包含多种英语阅读资料, 这些资料来自 LibriVox 项目上的英语有声读物, 音频内容全部为较长、含有多个自然句的文段. 可以看出, 实验采用的语音数据集包含了单词、中等句和长句 3 种不同语音类型, 可以有效验证提出方法是否在不同类型的语音数据集上都有一定的效果.

4.2 智能语音识别软件

实验采用 DeepSpeech 作为待测试的智能语音识别软件. DeepSpeech 是由百度开发的, 一种常用的端到端深度语音识别软件, 尤其在有噪声的环境中具有较好的语音识别效果. 而且该软件是通过 CTC 损失函数来计算预测的误差, 使用更为方便. 在实验中, 将每一轮迭代生成的语音对抗样本输入到 DeepSpeech 软件中, 输出当前对抗样本相应的文本内容, 将其与目标文本进行对比.

4.3 实验环境与衡量指标

实验所用系统环境为 Ubuntu 16.04 系统, 使用 Python 语言作为实验的编程语言, 使用的深度学习平台框架为

TensorFlow 1.12.

实验采用语音相似度、生成对抗样本所需时间以及生成对抗样本成功率这 3 个指标来衡量对抗样本生成方法的效果. 下面对这 3 个指标进行详细介绍.

(1) 语音相似度: 提出的方法通过在原始样本中不断添加噪声进行干扰, 最终生成对抗样本. 为了衡量加入噪声后的对抗样本与原始样本之间音频的差异, 需要知道二者音频方面的相似程度. 为此, 实验引入了皮尔逊相关系数 ρ, 来衡量对抗样本与原始样本间的语音相似度. 皮尔逊相关系数是专门用来衡量变量之间的线性相关程度的指标. 在本实验中, 相关公式[43]可以表示如下:

$$\rho_{x,x_{\text{adv}}} = corr(x, x_{\text{adv}}) = \frac{cov(x, x_{\text{adv}})}{\delta_x \delta_{\text{adv}}} \tag{5}$$

其中, cov 函数表示原始语音样本和对抗样本之间的协方差, δ 表示样本的标准差. 原始样本与对抗样本的语音相似度越高, 表明生成的对抗样本与原始音频样本越相似, 干扰噪声更具有隐蔽性.

(2) 生成时间: 生成目标对抗样本所需时间越短, 表明该方法在时间效率方面表现更为优异. 虽然可以通过比较各种算法的迭代次数来衡量生成效率, 但是考虑到本算法和对比方法的每一轮迭代所需时间都不相同, 简单使用迭代次数进行对比显然是不合理的, 本实验采用对抗样本所需的生成时间作为衡量指标. 在进行每一次生成对抗样本的实验时, 统计出开始时间 t_b 以及结束时间 t_e, 生成对抗样本所消耗的时间 t 就可以表示成 t_e–t_b.

(3) 成功率: 如果方法最终生成的对抗样本的文本内容和目标文本之间的编辑距离为 0, 就表明这种方法成功生成了目标对抗样本. 在选择的语音数据集上进行方法的效果评估时, 将在该数据集上成功生成目标对抗样本的实验次数表示成 a, 在该数据集上所做的所有实验次数表示成 b, 那么在该数据集上, 生成对抗样本的成功率 c 可以表示为:

$$c = \frac{a}{b} \times 100\% \tag{6}$$

4.4 实验对比方法

为了验证萤火虫-梯度对抗样本生成方法相比同类型生成方法是否具有优异的表现, 经过调研, 实验选择了 3 种同类型的对抗样本生成方法: Taori 等人[6]提出的遗传-梯度评估方法, Chen 等人[10]提出的布谷鸟算法以及没有进行调参优化的初始萤火虫算法, 来作为对比方法. 所使用的对比方法全部属于群智能算法, 即通过不同的方法构造出含有多个个体的种群, 然后在种群中不断寻找最优个体, 因而算法之间具有可比性. 在实验中, 对于同一个原始音频样本, 使用 4 种不同方法来生成目标对抗样本. 运用 3 种指标来衡量每种方法生成对抗样本的有效性和性能.

4.5 方法参数选择

在萤火虫算法中, 种群中个体数量 n, 随机搜索系数 α, 这些参数值的变化通常会让算法呈现出的效果千差万别. 因此, 实验前有必要为这些参数选择好合适的数值, 以保证在使用萤火虫算法时, 方法表现可以达到最优. 然而, 由于采用的原始音频样本数量比较多, 将其全部进行实验, 观察结果后再进行参数选择显然是不现实的. 因此, 实验在 3 种语音数据集上各挑选了 20 个语音数据样本, 针对 DeepSpeech 语音识别软件, 进行目标对抗样本的生成. 在生成中设置目标文本, 将优化方法的最大迭代次数设置为 3 000 次. 找到合适的参数之后, 继续完成剩余的样本生成.

种群中个体数量 n: 根据 Yang 等人[24]对萤火虫算法的研究, 种群个体数量 n 的取值在 25–40 之间最佳. 为了寻找最优的个体数量, 分别取 n 为 25、30、35、40, 系数 α 以 0.2 为单位, 分别取 α 为 0.2、0.4、0.6、0.8 和 1 这 5 个数值进行共 20 组实验, 输入挑选的音频样本, 计算出每种语音数据集上生成对抗样本时的成功率和语音相似度以及生成时间 3 种指标. 每种个体数量的实验取平均结果如图 3 和表 1 所示. 在图 3 中, 柱状图例代表成功率, 而折线代表对抗样本的语音相似度. 可以看出, 当 n 为 40 时, 3 种语音数据集上的成功率和语音相似度都是最高的, 表现最佳, 而从表 1 中看出, 当 n=40 时生成对抗样本所需时间虽然不是最短, 但是与所需最短时间相差不大. 在综合考虑下, 将种群中个体的数量 n 设定为 40.

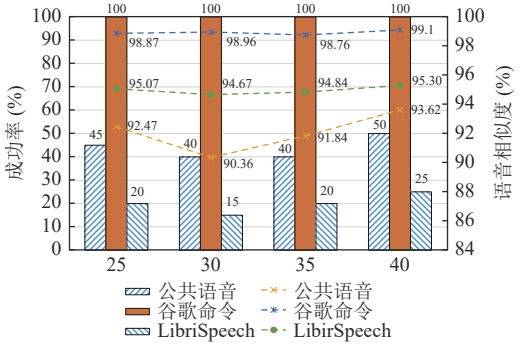

图 3 不同种群数量下的成功率与语音相似度

表 1 不同种群数量下各数据集对抗样本生成时间 (s)

种群大小	谷歌命令数据集	公共语音数据集	LibriSpeech语音数据集
25	94.5	2665.8	6 545.4
30	105.0	3076.2	7 367.3
35	130.5	2976.4	6 915.4
40	103.5	2765.6	6 741.6

系数α: 萤火虫算法中随机搜索系数α的取值范围在 0 至 1 之间, 以 0.2 为单位, 分别取α为 0.2、0.4、0.6、0.8 和 1 这 5 个数值. 在设置种群个体数量 n 分别为 25、30、35、40 的情况下, 与 5 种α取值进行共 20 组实验, 计算出每种语音数据集上 3 种衡量指标的数值. 每种随机搜索系数取值的实验取平均结果, 如图 4 和表 2 所示. 图 4 的柱状和折线分别代表生成样本的成功率和语音相似度. 可以看出, 当系数α取 0.2 的时候, 生成对抗样本的成功率和语音相似度都最高, 而此时生成样本所需的时间虽然不是最短, 但与最快生成时间相差不大. 经过综合考虑, 选择萤火虫算法的随机搜索系数α为 0.2.

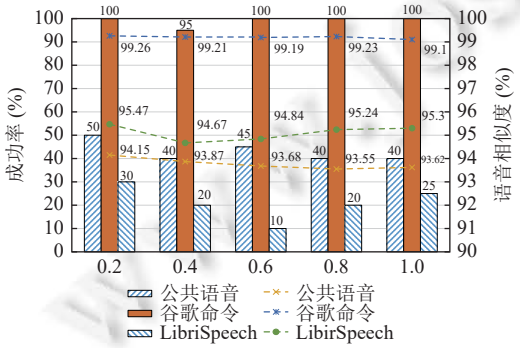

图 4 不同α取值下的成功率与语音相似

表 2 不同α取值下各数据集对抗样本生成时间 (s)

数值α	谷歌命令数据集	公共语音数据集	LibriSpeech语音数据集
0.2	104.0	2868.6	6867.4
0.4	133.7	3431.8	7284.2
0.6	115.5	3765.8	7168.1
0.8	106.0	3167.3	7441.5
1	103.5	2765.6	6741.6

5 实验结果与分析

为了评估萤火虫-梯度评估对抗样本生成方法的效果, 针对智能语音识别软件, 在选择的 3 种语音数据集上进行了 4 种对抗样本生成方法的实验评估. 其中, 将萤火虫-梯度评估方法中的种群个数设置为 40, 系数α的值设置为 0.2. 对于目标文本的设置, 将选取的公共语音数据集以及 LibriSpeech 语音数据集的目标文本设置成长度为 2 的常见短语, 例如“hello world”“jump out”等. 而对于谷歌命令语音数据集, 则将选择的每条命令任意生成其他 9 种不同命令和一个用户定义的单词命令. 例如, 在该语音数据集上选取了 go、stop、 left、right、on、off、 up、down、yes 和 no 这 10 个各不相同的命令, 将 go 命令分别生成其他 9 种命令和一个用户定义的 help 命令, 这样构造出该语音数据集上的 100 个样本的目标文本. 记录每一次实验时最终对抗样本与原始样本的语音相似度, 生成对抗样本所需时间以及是否成功生成目标对抗样本. 下面对 3 种指标上的实验结果进行详细分析.

5.1 语音相似度分析

在使用 4 种方法对原始音频样本进行实验、生成目标对抗样本后, 求出每一种方法最终生成的对抗样本与原始样本之间的平均语音相似度, 结果如表 3 所示.

从表 3 中可以看出, 在公共语音数据集和谷歌命令数据集上, 萤火虫-梯度评估方法的平均语音相似度数值只

略低于遗传-梯度评估方法, 而高于布谷鸟算法以及初始萤火虫算法. 而在 LibriSpeech 语音数据集上, 本方法的表现更为优异. 本文提出方法在 3 种不同类型的语音数据集上生成对抗样本的语音相似度都比较高, 均在 93% 以上. 这表明, 采用提出方法, 生成的目标对抗样本都和原始音频样本非常相似, 具有很好的隐蔽性.

表 3　4 种方法平均语音相似度比较 (%)

数据集	方法	DeepSpeech
谷歌命令	遗传-梯度评估方法	**99.10**
	布谷鸟算法	94.86
	初始萤火虫算法	98.26
	萤火虫-梯度评估方法	99.00
公共语音	遗传-梯度评估方法	**94.67**
	布谷鸟算法	92.31
	初始萤火虫算法	90.67
	萤火虫-梯度评估方法	93.14
LiberSpeech	遗传-梯度评估方法	93.45
	布谷鸟算法	92.91
	初始萤火虫算法	93.14
	萤火虫-梯度评估方法	**95.15**

为了对原始音频样本和对应的对抗样本二者之间的语音相似度有更为直观的了解, 实验在 3 种不同语音数据集上分别选取 2 个原始样本以及相应的目标对抗样本进行比较, 原始样本以 1-wav 至 6-wav 的方式按顺序进行命名, 相应的对抗样本以“adv-wav”为后缀进行命名.6 个原始样本的相关详细信息如表 4 所示. 其中, 挑选的原始音频文件 5-wav 以及 6-wav 二者的语音文本内容虽然相同, 但是由不同的志愿者所提供, 具有不同的发音效果. 因此, 二者属于不同的原始音频样本.

实验获取到每个原始样本与相应对抗样本的语音波形图, 对比结果如图 5–图 7 所示, 3 幅图中的音频样本分别来自谷歌命令数据集、公共语音数据集以及 LibriSpeech 数据集. 在波形对比图中, 横坐标代表时间, 纵坐标表示振幅, 采用两种不同的颜色分别表示原始样本和对抗样本的语音波形, 其中红色代表原始的语音波形, 而蓝色则代表加入噪声之后生成的目标对抗样本的语音波形.

表 4　部分原始样本的语音相似度

名称	原始样本内容	目标文本	语音相似度 (%)
1-wav	left	right	99.21
2-wav	right	left	97.65
3-wav	The boy was surprised	happy birthday	92.19
4-wav	Please had to move on	jump out	96.17
5-wav	this is a box recalling all box recalling remain in the public domain for our information to volunteer please visit start work	hello world	96.65
6-wav	this is a box recalling all box recalling remain in the public domain for our information to volunteer please visit start work	stay away	95.74

观察这 6 组语音波形对照图, 可以注意到: 几乎所有的扰动都添加在没有波形, 或者波形振动幅度较小的区域, 即样本中没有原始有效语音的区域, 将干扰噪声加入到这些区域中, 会让人们认为噪声来自原始样本的周围环境而加以忽略, 因而不易被察觉. 而在有效语音部分添加的噪声干扰很小, 几乎没发生变化, 因而保持了原始的有效语音内容, 不会影响正常的有效语音部分. 从 3 种不同语音数据集的角度来看, 可以更直观看出所加入噪声的差异: 在简单的谷歌命令数据集上, 只需要加入比较细微的噪声就可以生成对应的目标对抗样本, 而在 LibriSpeech 语音数据集中需要加入比较明显的噪声, 才能生成目标对抗样本. 当然这个结论并不是绝对的, 例如在略为复杂的公共语音数据集的原始样本 4-wav 上, 只需加入细微、几乎可忽略的干扰噪声, 就可以成功生成目标对抗样本.

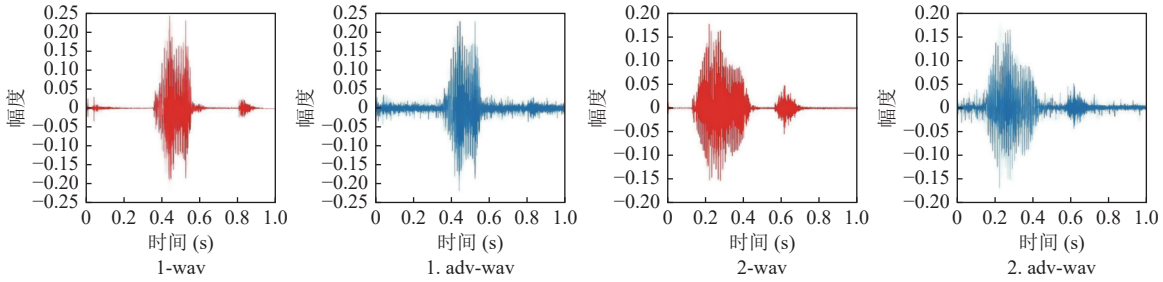

图 5 样本 1-wav、2-wav 的波形对比图

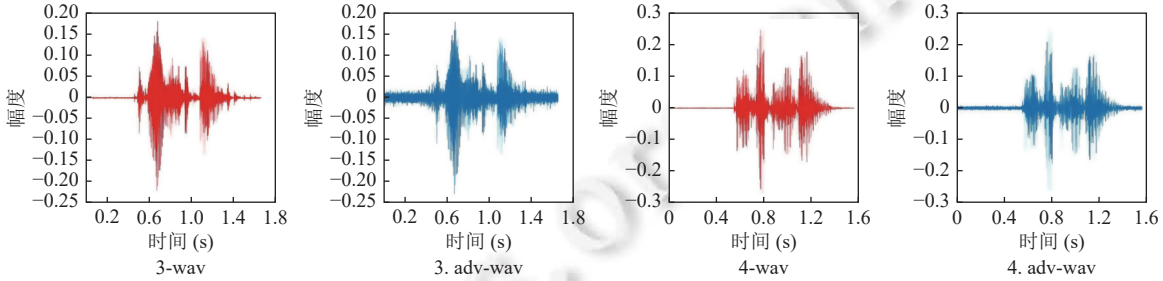

图 6 样本 3-wav、4-wav 的波形对比图

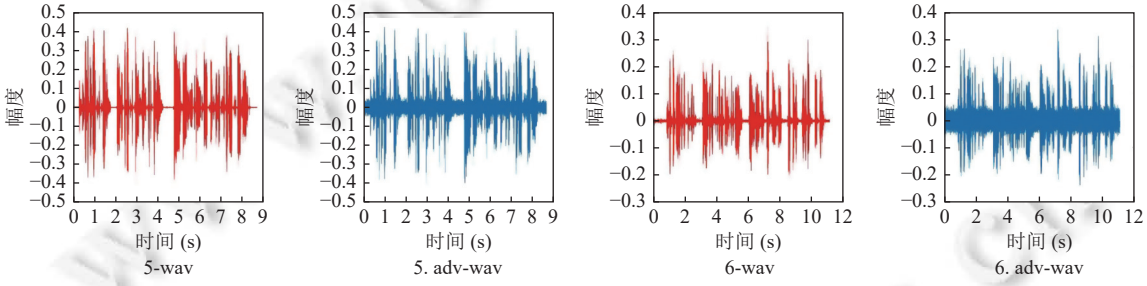

图 7 样本 5-wav、6-wav 的波形对比图

5.2 生成时间分析

本实验统计了 4 种方法生成目标对抗样本所需的平均时间, 结果如表 5 所示.

在 3 种语音数据集上, 萤火虫-梯度评估方法生成对抗样本所需时间都不是最短, 但是和其他 3 个方法所需的最短时间差距都不大, 且优于布谷鸟算法与初始萤火虫算法所需时间. 遗传-梯度评估方法通常需要很短的时间来生成目标对抗样本. 相比之下, 其他方法需要较长的时间才能在相较于遗传算法搜索范围更大的区域内找到最优解. 因此, 它们都比萤火虫-梯度评估方法所需时间长. 而从 3 种语音数据集的类型上看, 可以发现: 在原始音频样本中, 语音内容越短、越简单, 生成对抗样本所需时间越短, 例如在谷歌命令数据集上生成对抗样本平均时间只有两分多钟.

为了进一步分析对抗样本生成所需时间, 实验还选取了谷歌命令语音数据集上的部分评估结果, 如表 6 所示. 通过针对这些原始语音样本生对应目标的对抗样本, 能够观察到生成是否成功, 以及生成所需时间的不同. 例如, 为原始的“left”语音命令生成目标为“right”文本的对抗样本需要 142 s, 为原始“right”命令生成目标为“left”文本的对抗样本则需要 320 s, 而为原始“yes”语音命令无法成功生成目标为“right”文本的对抗样本. 同时也可以观察到, 谷歌命令语音的对抗样本生成所需时间相对较短, 有的甚至只需 19 s 就可以生成目标对抗样本. 在实际应用中, 如果对自动驾驶领域的原始语音样本进行干扰, 只需要比较短的时间就可将原始的“stop”命令和“no”命令最终转

换成“go”文本, 而在这段时间内人们却无法察觉到这种改变, 将对自动驾驶汽车的安全性和鲁棒性构成极大的威胁, 甚至导致交通事故的发生. 因此, 自动驾驶汽车要在噪声环境下进行充分的语音测试, 尤其要重视基于环境噪声干扰的测试研究.

表 5　4 种方法生成对抗样本时间 (s)

数据集	方法	DeepSpeech
谷歌命令	遗传-梯度评估方法	**125.6**
	布谷鸟算法	140.6
	初始萤火虫算法	162.5
	萤火虫-梯度评估方法	133.8
公共语音	遗传-梯度评估方法	**2 976.6**
	布谷鸟算法	3 521.8
	初始萤火虫算法	3 842.8
	萤火虫-梯度评估方法	3 167.3
LiberSpeech	遗传-梯度评估方法	**7 345.4**
	布谷鸟算法	7 727.3
	初始萤火虫算法	8 368.1
	萤火虫-梯度评估方法	7 541.6

表 6　部分样本生成目标对抗样本所需时间 (s)

原始样本内容	目标文本	生成对抗样本所需时间
left	right	142
right	left	320
yes	right	不成功
right	yes	200
go	stop	146
stop	go	60
go	no	19
no	go	25

5.3　成功率分析

经过实验, 最终得到 4 种对比方法在不同类型的语音数据集上生成目标对抗样本的成功率, 结果如表 7 所示. 成功率越高, 表示采用这种方法生成目标对抗样本的效果越好.

表 7　4 种方法成功率比较 (%)

数据集	方法	DeepSpeech
谷歌命令	遗传-梯度评估方法	92
	布谷鸟算法	92
	初始萤火虫算法	84
	萤火虫-梯度评估方法	**94**
公共语音	遗传-梯度评估方法	35
	布谷鸟算法	38
	初始萤火虫算法	32
	萤火虫-梯度评估方法	**48**
LiberSpeech	遗传-梯度评估方法	24
	布谷鸟算法	28
	初始萤火虫算法	24
	萤火虫-梯度评估方法	**34**

可以看出, 在 3 种语音数据集上, 萤火虫-梯度评估方法生成对抗样本的成功率均优于对照组的遗传-梯度评估方法、布谷鸟算法以及初始萤火虫算法, 在公共语音数据集和 LibriSpeech 语音数据集上, 本方法的成功率优势则更为明显. 这样的结果与算法自身的特点有关: 萤火虫算法的核心是利用种群间的吸引度和随机移动来不断改变种群中每个个体的位置, 从而搜索空间非常广阔, 因此生成对抗样本的成功率较高, 布谷鸟搜索算法和萤火虫算法类似, 也是通过广泛移动, 在广阔的搜索空间内寻优. 而遗传算法的核心是种群之间的选择、交叉、变异操作, 搜索空间相对比较小, 因此生成目标对抗样本的成功率不如前两者.

从不同的语音数据集的角度来看, 可以观察到, 提出的方法在包含单个单词的谷歌命令语音数据集上, 生成对抗样本成功率最高, 超过了 90%, 在包含中等长度句子的公共语音数据集上, 成功率相对较低, 而在包含长句的

LibriSpeech 语音数据集上生成对抗样本的成功率则最低. 一般来说, 在原始音频样本中, 语音内容越短、越简单, 生成目标对抗样本的成功率就越高; 反之, 原始样本语音内容越长、越复杂, 生成相应的对抗样本的成功率就越低.

由于针对语音智能软件的对抗样本生成比较复杂, 再加上针对的是参数未知的黑盒语音软件, 无法保证在任意情况下都可以成功生成目标对抗样本, 因此导致使用提出方法, 生成对抗样本成功率的绝对数值比较低. 然而, 和对比方法相比, 成功率已经大幅度提升, 例如在公共语音数据集上, 相比遗传-梯度评估方法将成功率从 35% 提升至 48%, 成功率提升了 13%.

5.4 人工验证样本

对抗样本生成方法生成的语音对抗样本应使得黑盒智能语音识别软件产生期望的错误识别, 而人们只能听到在原始音频样本中加入了一些细微干扰噪声, 无法察觉生成的对抗样本与原始样本的明显差异. 为了解所提方法生成的对抗样本是否能够使人们察觉不出与原始样本的明显差异, 还需要采用人工的方式对生成的对抗样本进行验证.

为此, 实验寻找了 30 名志愿者, 从 3 种语音数据集上分别任意挑选了 10 组原始样本以及成功生成的对应目标对抗样本. 这些志愿者都是大学学生, 此前并不了解对抗样本领域, 对于本实验所做研究也并不了解. 在进行人工验证时, 先给志愿者播放原始样本的音频内容, 然后再播放对应的目标对抗样本的音频内容. 每一组原始样本与目标样本播放结束后, 再告诉志愿者最终对抗样本转换成的目标文本, 并询问有没有听到目标文本的相关内容. 例如, 原始样本的音频内容为“left”, 目标文本为“right”, 最终需要询问志愿者有没有听到对抗样本里有关“right”的音频内容. 结果表明, 90% 的志愿者表示听到对抗样本和原始样本的音频内容一致, 只是前者存在一些细微噪声, 但仍能听清原始音频的内容, 并表示听不出任何有关目标文本的声音. 只有 10% 的志愿者表示 30 条对抗样本里只有一两组对抗样本中的噪声比较明显, 对原始的音频样本产生了干扰. 这表明采用提出方法生成的对抗样本, 能够让人们察觉不出与原始样本的明显差异. 然而, 仍然有一小部分对抗样本的噪声较为明显, 需要在之后的工作中减弱噪声.

5.5 结果分析

本节从原始样本与对抗样本的平均语音相似度、生成目标对抗样本所需时间以及生成对抗样本的成功率 3 个方面来对提出的萤火虫-梯度评估对抗样本生成方法以及选择的对比方法的实际效果进行评估. 结果表明, 虽然萤火虫-梯度评估对抗样本生成方法在生成对抗样本所需时间和平均语音相似度上的表现不是最好, 略低于遗传-梯度评估方法的表现, 然而, 在生成对抗样本的成功率方面, 提出的方法要优于其他 3 种对照方法, 尤其对于中等长度语句和长句而言更具优势. 实验还进行了人工验证, 表明生成的语音对抗样本是有效的.

另外, 实验将未经参数选择的初始萤火虫算法和萤火虫-梯度评估对抗样本生成方法进行对比, 可以发现后者在语音数据集上的综合性能表现远远优于前者. 这表明在实验之前对萤火虫算法进行重要参数的选择是十分必要的, 这样能够显著提高方法的效果.

6 总 结

针对黑盒智能语音软件, 本文采用了萤火虫算法与梯度评估相结合的方法, 在原始的音频样本中不断加入干扰噪声, 来生成目标对抗样本作为测试用例. 在使用的两种优化方法中, 萤火虫算法用于扩大寻优搜索范围, 而不是仅仅局限于某一固定空间进行寻优, 梯度评估方法则是当生成的对抗样本的文本内容和目标文本之间的编辑距离非常小时, 用来进行最终的局部优化. 实验表明本文提出的方法在 3 种语音数据集上都取得了一定的效果: 在语音相似度和对抗样本生成时间略差于对比方法的情况下, 目标对抗样本生成的成功率在不同类型的数据集上均优于对比方法.

然而, 提出的萤火虫-梯度评估对抗样本生成方法也存在一些不足之处: 生成目标对抗样本的成功率虽然比之前提高, 但仍有很大的提升空间; 原始音频和生成的对抗音频的语音相似度相对比较低. 在后面的工作中, 可以采

用其他优化方法, 在保持样本语音相似度和生成样本所需时间比较合理的情况下, 能够将生成对抗样本的成功率进一步提升.

References:

[1] Cook DJ, Crandall AS, Thomas BL, Krishnan NC. CASAS: A smart home in a box. Computer, 2013, 46(7): 62–69. [doi: 10.1109/MC.2012.328]

[2] Sermanet P, LeCun Y. Traffic sign recognition with multi-scale convolutional networks. In: Proc. of 2011 Int'l Joint Conf. on Neural Networks. San Jose: IEEE, 2011. 2809–2813. [doi: 10.1109/IJCNN.2011.6033589]

[3] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Proc. of the 3rd Int'l Conf. on Learning Representations. San Diego: ICLR, 2015.

[4] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow IJ, Fergus R. Intriguing properties of neural networks. In: Proc. of the 2nd Int'l Conf. on Learning Representations. Banff: ICLR, 2014.

[5] Wang DH, Wang RD, Dong L, Yan DQ, Zhang XY, Gong YK. Adversarial examples attack and countermeasure for speech recognition system: A survey. In: Proc. of the 1st Int'l Conf. on Security and Privacy in Digital Economy. Quzhou: Springer, 2020. 443–468. [doi: 10.1007/978-981-15-9129-7_31]

[6] Taori R, Kamsetty A, Chu B, Vemuri N. Targeted adversarial examples for black box audio systems. In: Proc. of the 2019 IEEE Security and Privacy Workshops (SPW). San Francisco: IEEE, 2019. 15–20. [doi: 10.1109/SPW.2019.00016]

[7] Chen JY, Ye LH, Zheng HB, Yang YT, Yu SQ. Black-box adversarial attack toward speech recognition system. Journal of Chinese Computer Systems, 2020, 41(5): 1019–1029 [doi: 10.3969/j.issn.1000-1220.2020.05.020]

[8] Khare S, Aralikatte R, Mani S. Adversarial black-box attacks on automatic speech recognition systems using multi-objective evolutionary optimization. In: Proc. of the 20th Annual Conf. of the Int'l Speech Communication Association. Graz: ISCA, 2019. 3208–3212. [doi: 10.21437/Interspeech.2019-2420]

[9] Yang XS. Firefly algorithm, stochastic test functions and design optimisation. Int'l Journal of Bio-inspired Computation, 2010, 2(2): 78–84. [doi: 10.1504/IJBIC.2010.032124]

[10] Mozilla. Common Voice Dataset. 2017.

[11] Warden P. Speech commands: A dataset for limited-vocabulary speech recognition. arXiv: 1804.03209, 2018.

[12] Panayotov V, Chen GG, Povey D, Khudanpur S. Librispeech: An ASR corpus based on public domain audio books. In: Proc. of the 2015 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP). South Brisbane: IEEE, 2015. 5206–5210. [doi: 10.1109/ICASSP.2015.7178964]

[13] Baum LE, Petrie T, Soules G, Weiss N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. The Annals of Mathematical Statistics, 1970, 41(1): 164–171. [doi: 10.1214/aoms/1177697196]

[14] Yu D, Deng L, Dahl G. Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition. In: Proc. of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning. 2010.

[15] Mohamed AR, Dahl GE, Hinton G. Acoustic modeling using deep belief networks. IEEE Trans. on Audio, Speech, and Language Processing, 2012, 20(1): 14–22. [doi: 10.1109/TASL.2011.2109382]

[16] Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks. In: Proc. of the 31st Int'l Conf. on Machine Learning. Beijing: JMLR. org, 2014. 1764–1772.

[17] Sohn J, Kim NS, Sung W. A statistical model-based voice activity detection. IEEE Signal Processing Letters, 1999, 6(1): 1–3. [doi: 10.1109/97.736233]

[18] Muda L, Begam M, Elamvazuthi I. Voice recognition algorithms using Mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. arXiv: 1003.4083, 2010.

[19] Itakura F. Line spectrum representation of linear predictor coefficients of speech signals. The Journal of the Acoustical Society of America, 1975, 57(S1): S35. [doi: 10.1121/1.1995189]

[20] Hermansky H. Perceptual linear predictive (PLP) analysis of speech. The Journal of the Acoustical Society of America, 1990, 87(4): 1738–1752. [doi: 10.1121/1.399423]

[21] Graves A, Fernández S, Gomez F, Schmidhuber J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: Proc. of the 23rd Int'l Conf. on Machine Learning. Pittsburgh: ACM, 2006. 369–376. [doi: 10.1145/1143844.1143891]

[22] Yang XS. Nature-inspired Metaheuristic Algorithms. Frome: Luniver Press, 2010.

[23] Yang XS, He X. Firefly algorithm: Recent advances and applications. Int'l Journal of Swarm Intelligence, 2013, 1(1): 36–50. [doi: 10.1504/IJSI.2013.055801]

[24] Yang XS. Firefly algorithms for multimodal optimization. In: Proc. of the 5th Int'l Symp. on Stochastic Algorithms. Sapporo: Springer, 2009. 169–178. [doi: 10.1007/978-3-642-04944-6_14]

[25] Liszka T, Orkisz J. The finite difference method at arbitrary irregular grids and its application in applied mechanics. Computers & Structures, 1980, 11(1–2): 83–95. [doi: 10.1016/0045-7949(80)90149-2]

[26] Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometrics and Intelligent Laboratory Systems, 1987, 2(1–3): 37–52. [doi: 10.1016/0169-7439(87)80084-9]

[27] Masek WJ, Paterson MS. A faster algorithm computing string edit distances. Journal of Computer and System Sciences, 1980, 20(1): 18–31. [doi: 10.1016/0022-0000(80)90002-1]

[28] Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A. The limitations of deep learning in adversarial settings. In: Proc. of the 2016 IEEE European Symp. on Security and Privacy (EuroS&P). Saarbruecken: IEEE, 2016. 372–387. [doi: 10.1109/EuroSP.2016.36]

[29] Kurakin A, Goodfellow IJ, Bengio S. Adversarial examples in the physical world. In: Proc. of the 5th Int'l Conf. on Learning Representations. Toulon: OpenReview.net, 2017.

[30] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: Proc. of the 2017 IEEE Symp. on Security and Privacy (SP). San Jose: IEEE, 2017. 39–57. [doi: 10.1109/SP.2017.49]

[31] Carlini N, Wagner D. Audio adversarial examples: Targeted attacks on speech-to-text. In: Proc. of the 2018 IEEE Security and Privacy Workshops (SPW). San Francisco: IEEE, 2018. 1–7. [doi: 10.1109/SPW.2018.00009]

[32] Cisse M, Adi Y, Neverova N, Keshet J. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In: Proc. of the 31st Annual Conf. on Neural Information Processing Systems. Long Beach, 2017. 6980–6990.

[33] Schönherr L, Kohls K, Zeiler S, Holz T, Kolossa D. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In: Proc. of the 26th Annual Network and Distributed System Security Symp. San Diego: The Internet Society, 2019. [doi: 10.14722/ndss.2019.23288]

[34] Qin Y, Carlini N, Cottrell G, Goodfellow I, Raffel C. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 5231–5240.

[35] Alzantot M, Balaji B, Srivastava M. Did you hear that? Adversarial examples against automatic speech recognition. In: Proc. of the NIPS 2017 Machine Deception Workshop. 2017.

[36] Du TY, Ji SL, Li JF, Gu QC, Wang T, Beyah R. SirenAttack: Generating adversarial audio for end-to-end acoustic systems. In: Proc. of the 15th ACM Asia Conf. on Computer and Communications Security. ACM, 2020. 357–369. [doi: 10.1145/3320269.3384733]

[37] Zhang GM, Yan C, Ji XY, Zhang TC, Zhang TM, Xu WY. DolphinAttack: Inaudible voice commands. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 103–117. [doi: 10.1145/3133956.3134052]

[38] Hannun A, Case C, Casper J, Catanzaro B, Diamos G, Elsen E, Prenger R, Satheesh S, Sengupta S, Coates A, Ng AY. DeepSpeech: Scaling up end-to-end speech recognition. arXiv: 1412.5567, 2014.

[39] Povey D, Ghoshal A, Boulianne G, Burge L, Glembek O, Goel N, Hannemann M, Motlicek P, Qian Y, Schwarz P, Silovsky J, Stemmer G, Vesely K. The Kaldi speech recognition toolkit. In: Proc. of the 2011 IEEE Workshop on Automatic Speech Recognition and Understanding. IEEE, 2011.

[40] Gong Y, Li BY, Poellabauer C, Shi YY. Real-time adversarial attacks. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence. IJCAI.org, 2019. 4672–4680.

[41] Esmaeilpour M, Cardinal P, Koerich AL. Towards robust speech-to-text adversarial attack. In: Proc. of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence. 2020. 3–10.

[42] Reichenbach T, Hudspeth AJ. Discrimination of low-frequency tones employs temporal fine structure. PLoS One, 2012, 7(9): e45579. [doi: 10.1371/journal.pone.0045579]

[43] Benesty J, Chen JD, Huang YT, Cohen I. Pearson correlation coefficient. In: Cohen I, Huang YT, Chen JD, Benesty J, eds. Noise Reduction in Speech Processing. Berlin, Heidelberg: Springer, 2009. 1–4. [doi: 10.1007/978-3-642-00296-0_5]

附中文参考文献:

[7] 陈晋音, 叶林辉, 郑海斌, 杨奕涛, 俞山青. 面向语音识别系统的黑盒对抗攻击方法. 小型微型计算机系统, 2020, 41(5): 1019–1029. [doi: 10.3969/j.issn.1000-1220.2020.05.020]

袁天昊(1997－), 男, 硕士生, 主要研究领域为人工智能软件测试, 图像检索.

吉顺慧(1987－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为软件建模, 测试与验证.

张鹏程(1981－), 男, 博士, 博士生导师, CCF 高级会员, 主要研究领域为人工智能软件测试, 服务计算, 数据科学.

蔡涵博(1995－), 男, 博士生, CCF 学生会员, 主要研究领域为数据安全, 人工智能软件测试.

戴启印(1996－), 男, 硕士生, 主要研究领域为人工智能软件测试.

叶仕俊(1996－), 男, 硕士生, 主要研究领域为人工智能软件测试.

任彬(1997－), 男, 硕士生, 主要研究领域为人工智能软件测试.