

# 基于声纹与人脸双重识别的考勤系统

## ——多媒体信息处理实验报告

葛奇昱 林佳豪 刘鑫宇 舒子骏 张博为

课堂考勤是当代大学课堂不可缺失的一环，但现有的传统考勤方式却常常因为点名时间长、识别精度低遭到诟病。半自动化与自动化的考勤方式具有巨大的应用前景，但是单纯基于面部识别的考勤方法常常会面临因为相机位置不佳导致人脸大量遮挡、镜头分辨率低难以提取人脸特征以及摄像头视野不够难以涵盖全部场景等问题，导致识别的精确率降低、识别周期增长。而单纯基于声纹的检测方法则难以高精度应对高噪声环境下的识别问题。为了提高考勤效率并克服基于单纯人脸识别技术或声纹识别技术的考勤系统的弊端，我们团队开发了一种基于声纹与图像双重识别的考勤程序，制作了便于用户交互的 UI 界面，并实现了本地检测与实时检测两种方法。

由于小组作业常常会因为数据缺失难以训练模型，我们团队选择使用基于深度学习的预训练模型，积极探索了 **few-shot** 技术，并借助孪生网络增大了不同样本特征间的区分度，在较好地克服模型泛化能力不足问题的同时提高了检测精度。此外，为了实现声纹与图像双重识别的目标，我们团队选择先从声纹识别与图像识别两个子问题展开，并尝试了从结果层面融合与特征层面融合两种方式探索性能更佳的识别方法。

下面将一一介绍我们项目的各个部分所面临的问题以及我们经过方法调研最终选取的解决方案。

**Web UI** 方面，所面临的主要问题是设计不同任务对应的用户交互界面以及与主函数的交互逻辑。具体而言，**Web UI** 要实现三个目标：样本输入——用户输入待检测人员（如学生）的照片、声音与姓名至 UI 界面，UI 界面将样本信息传给主函数并给出样本输入的信号，当主函数处理完成后将信号反馈给 UI 呈现给用户；本地视频检测——用户选择从本地上传视频或通过电脑内置的摄像头与麦克风进行录制，UI 将视频传递给主函数进行处理，主函数将处理结果，即到场人员名单和缺勤人员名单以字典的形式返回给 UI 并输出给用户；实时检测——基本逻辑与本地视频检测相近，区别在于其需要处理流式的输入并以流式的形式将点到的音频输出到用户界面，这种异步逻辑的处理有一定的难度，体现了研究的创新性与复杂性。

为了实现这个前端开发任务，我们尝试了两种 **Web UI** 开发软件。其一是使用由饿了么团队开发的基于 **Vue** 的 **Element UI** 开发软件，其优势是产品可以直接作为 **app** 嵌入移动设备使用，但难点是前置知识较多且与主程序交互较为复杂。其二是使用可以在 **python** 中导入的 **gradio** 库进行开发。其优势在于与 **python** 程序交互方便并且易于上手，缺陷则在于难以作为 **app** 集成到移动设备。最终，为了接口易于交互以及调试方便，我们团队选择使用 **gradio** 作为 UI 开发的媒介。



图 1: gr.Blocks()与 gr.Interface()框架下的页面设计

而在 `gradio` 中，有两种 Web UI 开发方式：其一是使用 `gradio.Interface()` 函数作为基础，通过对其设置参数的方法开发设计页面；其二是使用 `gr.Blocks()`，以块状、类似网页开发的方式进行 UI 开发。前者的好处在于功能集成度高，尤其对于流式输入输出的实现较为简单；但劣势也非常明显，难以处理多事件触发的逻辑。而后者的实现则较为复杂，并且函数的输入输出逻辑需要精心设计，但在实现多事件的触发与监听方面灵活性更大。为了实现样本输入、本地视频检测以及实时检测的复杂逻辑，我们选择使用 `gr.Blocks()` 作为 UI 设计的框架。

就技术方案而言，首先，我们确定了 `gradio` 的块层次代码框架，通过对各个组件的协调排版构建出整体的网页格局。其次，通过分页将各个任务的监听逻辑进行了较好的区分，降低了逻辑处理的复杂性。最后，通过监测按钮的状态、内置状态的变化以及流式输入的信息调度相关函数运转，最终将信息反馈给主函数进行处理，并接受返回值输出到用户界面。这样便实现了用户与程序的交互。

接下来通过介绍流式处理部分，来一窥整个项目所需的模型及主程序的逻辑：

首先，因为我们选择的是 `gradio` 框架，而 `gradio` 无法直接实现实时的流式效果，只能把音频流和图像流切分为无数多个小部分，分别传入函数运行，以达到近似实时的效果，这要求程序的效率足够高。又由于对输入数据的处理分为对图像的人脸检测与对音频的语音检测两部分，为避免其中一个线程的堵塞造成整个函数的堵塞，从而不符合我们预期的实时检测效果的情况，我们决定使用异步处理操作来提高整个程序的效率，以达到近似实时的效果。接下来将围绕下面的流程图详细介绍一下流式处理的逻辑过程：

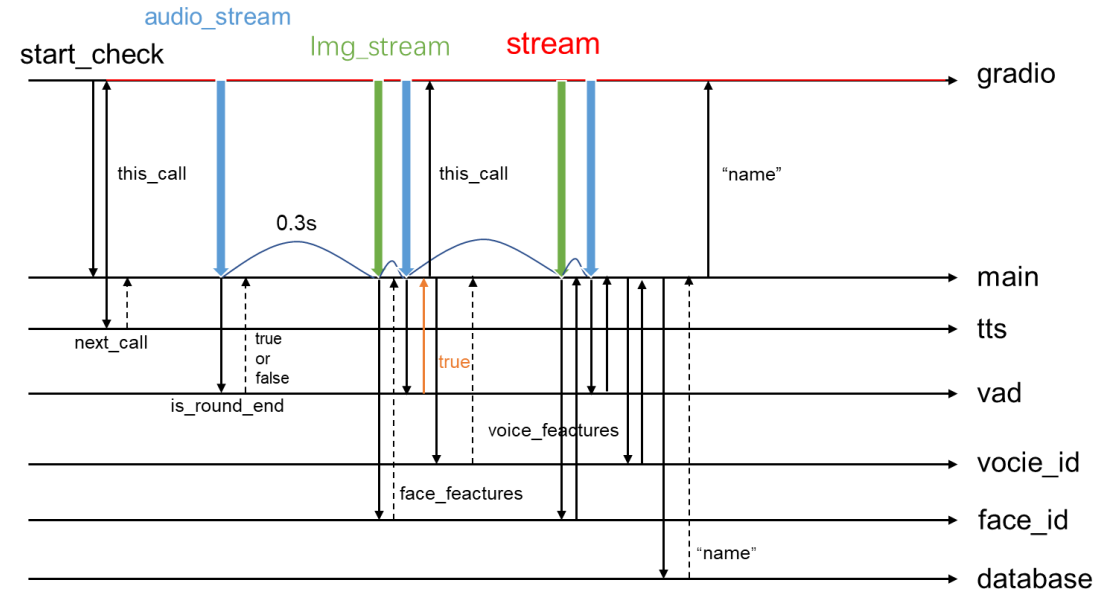


图 2：流式处理调度流程图

根据我们设想的自动点名过程，程序首先根据点名表生成音频返回给 `gradio` 界面，`gradio` 界面检测到有视频输入时自动播放点名。当被点名者答“到”时，`vad` 模型识别到一轮问答结束，返回下一个人的点名音频，以此循环，直至点名表上的所有人都被点名。

在此基础上，我们整体的流式处理过程如下：程序启动时，将初始化生成第一个人的点到音频“this call”。当用户点击 `gradio` 界面的“开始检测”按钮，程序会监听此事件并返回“start\_check”的 mode 信号给处理流式的函数 `handle_stream()`，其会返回 `this call` 给 `gradio` 界面自动播放，并启动流式（stream）输入与处理。在流式输入模式下，将以 0.3 秒为一个 chunk 输入音频流，持续 3 秒；在此之后，将以 0.01

秒为一个 chunk 输入图像流，持续 0.01 秒，以此循环。在实际使用中，可以根据音频流与图像流的处理情况对这些参数进行调整，从而获得最佳的识别性能。

在流式处理中，每次主函数接收音频流，便会将音频传给 vad 和 voice\_id 模型，并为 vad 模型创建一个进程用于判断一轮问答是否结束。与此同时，检索该进程下有没有已完成的任務，如果有，并且任务返回的结果是“True”，便为 voice\_id 模型创建一个新进程用作提取音频特征向量，并生成 next\_call 用作点下一个人的名字；而如果结果是“False”，则维持其他功能的运行。而每次接受图像流时，主函数将图像传给 face\_id 模型，并为 face\_id 模型创建一个进程用作提取人脸特征向量。无论接受音频流还是图像流，最后都会检索提取特征向量的进程是否有已完成的任務，如果有，为 database 模型创建一个新进程用作识别特征向量，返回已经到场的名单，主程序将到场人员的姓名以字典的形式返回给 gradio 显示；如果没有，则结束，等待 gradio 的下次调用。经过我们调试，发现不太需要 tts 来帮助点名，最终将该进程删去，优化代码。

设想是美好的，现实是骨感的。在程序运行时却发现一些模型是不可序列化的，因此只能换种思路。考虑到 python 在 GPU 没有 gil（全局解释器锁），所以利用多线程仍然可以在 GPU 上实现并行任务。同时为了避免任务被识别成协程而报错，我们通过 asyncio 库把同步函数包装成异步函数，最终程序成功运行。经实战检验，我们使用的模型都算轻量化的模型，实际运行起来并不耗多少时间，即使使用同步程序，也一样能够实现近似实时检测的效果。为了代码的简洁性，我们最终把异步过程做成了同步函数并实现了预期效果。

下面将从方法调研、技术方案以及研究总结三个方面对我们研究主体的技术细节进行介绍：

## 一. 方法调研

### 1. 声纹识别

为了实现基于声纹的说话人识别，我们探索了许多较为使用的技术手段。

- **传统方法** [https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice\\_id.md](https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice_id.md)

声纹识别技术最早可以追溯到 20 世纪 60 年代。在传统技术时代，研究人员依赖于声学特征如线性预测倒谱系数（LPC）、感知线性预测系数（PLP）和梅尔频率倒谱系数（MFCC）来提取声音特征。这些特征通过动态时间规整（DTW）和矢量量化（VQ）等模板匹配方法进行分类。

2000 年后，基于高斯混合模型（GMM）和通用背景模型（UBM）的技术成为主流。这种方法通过统计特征建模实现了更高的识别精度，随后引入的联合因子分析（JFA）和 i-vector 技术进一步改进了模型的鲁棒性和效率。

- **深度学习** [https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice\\_id.md](https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice_id.md)

近年来，深度学习在声纹识别中的应用为该领域带来了质的飞跃。基于深度学习的声纹识别方法突破了传统技术的局限性，特别是在高噪声环境和域不匹配情况下表现出色。深度学习的核心优势在于其强大的特征表达能力，可以从语音信号中提取高度抽象的嵌入特征。

深度学习模型（如 DNN、CNN 和 RNN）能够从时间域或频谱域的输入中提取更具辨识力的特征，如 d-vector 和 x-vector。特别是 x-vector 方法通过引入统计池化层，从帧级特征提取段级嵌入，在多语种、短语音测试中取得了突破性成果。端到端的声纹识别方法直接从语音波形中学习特征并进行分类。这种方法避免了传统阶段式处理的复杂性，利用联合优化的方式提高了整体性能。





- **facenet**

该方法有 128 维和 512 维两种版本，由于后来发现 GitHub 上的 facenet-pytorch 仅提供了 512 维版本，因此在两种版本间对比不在此赘述。其测试效果极佳，当相似度阈值取到 0.7 时，无论是正常情况还是都坐在后排的情况都能较快地检测出所有人脸（后排情况下露出脸的只有 4 人）。




图 4: facenet 在后排情况下的检测结果-1

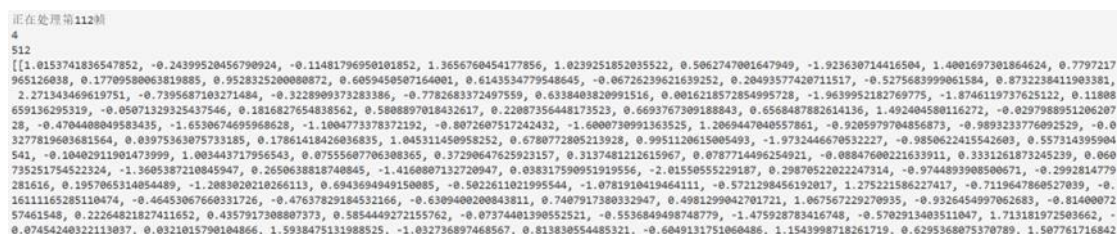


图 5: facenet 在后排情况下的检测结果-2

- **arcface**

实际上这种方法也相当不错，对于正常情况，相似度阈值只需取到 0.6 即可迅速顺利完成五张人脸的检测。由于我选择的策略是放宽相似度阈值以应对可能存在的复杂情况，因此得到正确结果所需的阈值越低调整空间越大，在这方面 arcface 效果不错。不过在面对人都在后排的情况时，如果不调节阈值，检测效果会略逊于 facenet，在 0.6 阈值下无法完整进行检测。

总体而言两种方法可以说是相当的。但 arcface 在 GitHub 上的项目中例子以及教程都比较简短，而我作为初学者比较担心能否顺利在短时间内解决遇到的问题；而 facenet 的项目中不仅有许多例子，而且 readme 文件和各种例子都有官方找人翻译的中文版本，非常详尽，因此其实让我放弃 arcface 而选择 facenet 的主要原因是这个。

- **openface**

这种方法效果并不好，对于正常情况，阈值选择 0.7 时无法完整检测，而阈值选择 0.8 又会远远超出正确人数，无论是因为对阈值敏感还是本身效果不好，都不太适用于本次任务，因此选择放弃了。

### 3. 数据存储、孪生网络训练与识别

在这个项目中，主要目标是利用对输入的脸部特征向量 `face_feature_vector` 和语音特征向量 `voice_feature_vector` 进行高效、准确的识别，以适应项目服务于课堂点名的应用环境。随着切身实地地感受代码编程的过程和任务的深入，我们尝试了从传统机器学习算法到深度学习网络的多种方法，并经历了从拼接特征向量到分开，从 knn 到

SVM、随机森林再到 Siamese Network 孪生网络的开发历程。以下是开发过程中不同阶段的思路 and 探索。

- **第一阶段：特征向量拼接与 KNN 分类识别**

最初，我们将脸部和语音特征向量直接拼接，形成一个联合特征向量，并使用 K 最近邻（KNN）算法进行分类。

这种方法简单易实现，因为 KNN 直接利用特征空间的欧几里得距离进行分类，无需复杂的模型训练，也无需大量数据支持。且作为结构简单的传统算法，KNN 推理速度快，计算代价小，项目易于调整。

但是，特征融合缺乏理论支持：脸部特征（512 维）与语音特征（192 维）之间的分布不同，直接拼接导致高维特征向量对分类性能的贡献不平衡。此外，KNN 对数据量较为敏感，当特征库增大时，查询速度显著下降。而且 KNN 所应用的欧几里得距离难以适应非线性分布。

- **第二阶段：分开处理特征向量，保留 KNN**

针对拼接特征带来的问题，我们尝试将脸部特征向量和语音特征向量分开处理，分别训练两个独立的 KNN 模型并应用于识别任务中。最后，综合两个模型的分类结果进行最终预测。

然而，KNN 模型过于简单的劣势明显放大：虽然分开处理特征提高了一些性能，但 KNN 本质上的线性假设仍难以适应复杂特征分布。参数调整所带来的分类性能提升有限。

- **第三阶段：SVM 和随机森林的尝试**

为了进一步提高识别性能，我们分别尝试了支持向量机（SVM）和随机森林算法来处理脸部特征和语音特征。这两个算法是传统机器学习中的经典非线性分类器，能够处理更复杂的分布。

其中，SVM 适合处理高维特征（如脸部特征向量），特别是当类别边界清晰时，使用核函数能更好地拟合非线性分布。

而选用随机森林来处理语音特征则是由于决策树集成能够捕捉特征与类别之间的非线性关系，且随机森林对噪声和数据缺失具有较强的鲁棒性，更适用于包含更多随机性的语音特征。

但是，两种传统方法分别应用显然实现不了脸部和语音联合识别的初衷。并且 SVM 与随机森林的内存消耗大，前者对噪声敏感，后者又对特征相关性敏感，在随机生成特征向量数据下表现堪忧。

那么，有没有一种结构清晰的模型，能够同时应用于两类特征向量的识别，且在识别过程中有着对数据的强适应性和识别结果的高准确性？

- **第四阶段：孪生网络（Siamese Network）**

在传统机器学习方法的局限性下，我们最终转向深度学习，选择了孪生网络（Siamese Network）用于脸部和语音特征向量的识别。孪生网络通过学习特征向量间的相似性来进行分类，更适合高维特征向量的匹配任务；同一网络结构框架下可以分别训练脸部特征和语音特征的相似性模型，避免了独立模型间的融合难题。

## 二. 技术方案

## 1. 声纹识别 [https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice\\_id.md](https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice_id.md)

本项目涉及音频-特征-文字处理的任务有：本地视频检测模式下，将视频音频利用语音活动检测按一轮问答（一次点到-答到）为单位进行切分，然后分离出答到部分音频，随后提取声纹特征；实时检测模式下，利用文本转语音按照名单产生点名音频，待点名音频播放结束后，通过语音活动检测持续监测是否有人答到，当检测到有人答到或是超过最长等待时间，对本轮答到音频进行声纹特征提取。

### ● 语 音 活 动 检 测 (VAD) [https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice\\_id.md](https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice_id.md)

本项目采用了 Silero VAD 模型进行语音活动检测。Silero VAD 是一个预训练的企业级语音活动检测器，能够高效、准确地识别音频中的语音片段。其主要特点包括：

- ❖ **高精度**：在语音检测任务中表现出色，能够有效区分语音与非语音部分。
- ❖ **低延迟**：处理单个音频块（30 毫秒以上）在单个 CPU 线程上耗时不到 1 毫秒，支持实时应用需求。
- ❖ **轻量级**：模型大小约为两兆字节，便于在资源受限的环境中部署。
- ❖ **通用性**：支持 8000 Hz 和 16000 Hz 采样率，适用于不同领域和背景噪声的音频。

在本地视频检测模式下，Silero VAD 被用于将视频音频按一轮问答（一次点到-答到）为单位进行切分，分离出答到部分音频。Silero VAD 的高精度和低延迟为系统的实时性能提供了重要保障。在实时检测模式下，我们采用传统的阈值方法，先得到音频的包络，如果其绝对值超过了某一阈值，就判定为已经答到。

### ● 声 纹 特 征 提 取 ( Speaker Embedding ) [https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice\\_id.md](https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice_id.md)

本项目采用 ECAPA-TDNN 模型进行声纹特征提取。ECAPA-TDNN (Emphasized Channel Attention, Propagation and Aggregation Time Delay Neural Network) 是一种先进的说话人识别模型，能够从音频数据中提取高维嵌入向量，用于识别和验证答到者的身份。

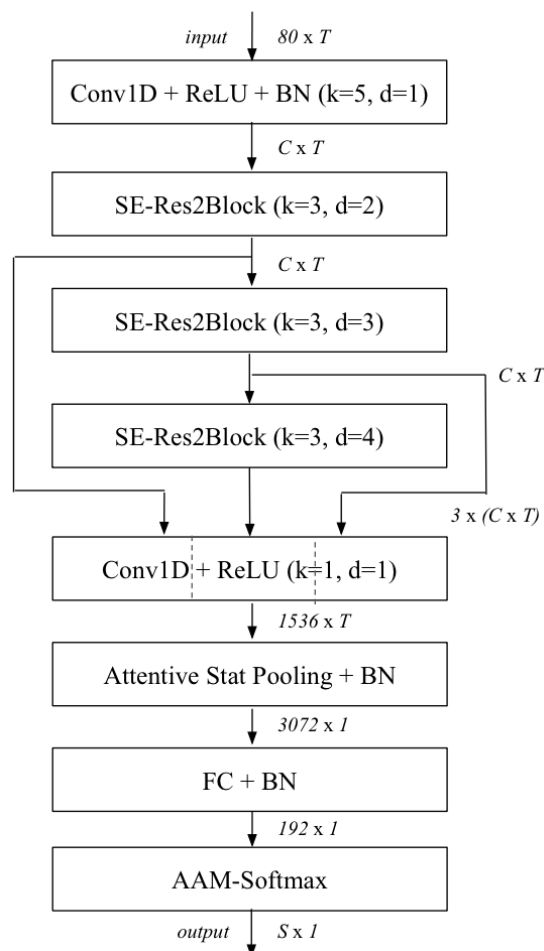


图 6: ECAPA-TDNN 网络结构

ECAPA-TDNN 模型的主要特点包括:

- ❖ 增强的通道注意力机制: 通过引入 SE-block (Squeeze-and-Excitation Block), 增强了模型对不同通道特征的关注能力, 提高了特征提取的精度。
- ❖ 残差连接和多尺度特征聚合: 采用 Res2Block 结构, 实现多尺度特征的聚合, 捕获更多的语音特征信息。
- ❖ 注意力统计池化: 使用 Attentive Statistical Pooling 方法, 对特征进行加权聚合, 提升了模型对变长语音输入的处理能力。

在本项目中, ECAPA-TDNN 模型被用于从音频中提取说话人嵌入向量, 这些嵌入向量作为唯一的语音特征标识, 用于后续的身份验证。该模型具有高度的准确性和良好的抗噪能力, 能够应对复杂的声学环境, 确保系统在各种场景下的可靠性。

## 2. 人脸识别

facenet 方法功能较为完善, 并且配有多种预训练模型。

具体到该任务上, 我使用了其内部的 mtcnn 深度学习模型进行人脸检测, Inception ResNet (V1) 深度卷积神经网络架构进行人脸特征提取, 同时为了解决数据缺少的问题, 我使用了其自带的在 vggface2 上训练的预训练模型。在使用 `get_features_list` 函数得到每帧图像的特征列表后, 再使用 `is_new_feature` 函数对得到的特征进行检测, 根据余弦相似度判断是否是全新的人脸特征, 并对现有的人脸特征列表进行维护更新。

为提高提取效率, 在 `get_features_list` 函数中我对每张图片中的人脸进行了批量



化处理，将所有人脸作为一个 batch 进行特征提取，减少了 Inception ResNet (V1) 网络的使用次数。

在进入实际运行时，发现有如下问题：一、观看测试过程的视频，发现时不时会在完全没有人的地方检测出一个人脸框；二、有些并不好的脸部（比如正在侧身走动的人）得到的向量也会被加入，而只要每一个人有一个好的正脸特征向量即可用于识别了。因此我实现了一个过滤有效人脸的功能。



图 7：正确的人脸检测实例



图 8：错误的人脸检测实例



图 9：侧脸检测实例

由于根据视频显示，这些错误不会连续出现，而且通常远小于人脸框，因此我通过



❖ **数据存储量改变：**one-shot learning->few-shot learning，即同一人的（姓名，脸部特征向量，语音特征向量）转换成（姓名，脸部特征向量列表，语音特征向量列表）。通过存储多个特征向量，避免单个特征使用时受到外界和本人状态影响，增强鲁棒性和准确性。且在孪生网络训练时提升训练效果。

❖ **Siagmese 网络模型增加处理步骤：**从两个全连接层中增加一个 Dropout 层并添加归一化步骤。Dropout 的作用：通过随机屏蔽部分神经元，减少过拟合，提高模型的鲁棒性；归一化的作用：使训练更稳定、收敛更快，特别是在处理高维度特征时防止梯度消失或爆炸。

❖ **自动计算阈值：auto\_threshold**

代码实现了提取类原型：从所有学生对象中提取对应的人脸或语音的特征向量。然后是计算类间相似性：利用余弦相似度，计算所有类别原型之间的相似性分数，得到类间距离集合。接着便是动态阈值调整：根据最大类间距离，结合权重参数 alpha，动态计算合适的识别阈值，并输出结果。

整体代码通过自动化和动态化阈值设定，有效减少手动调整的复杂性，同时适应不同任务场景的数据特性，从而提升模型在特征识别任务中的鲁棒性和准确性。

### 三. 效果演示与结果分析

生成网站如图所示：



分为三个界面，分别是“输入样本”、“视频检测：本地视频版”、“视频检测：实时检测版”，接下来将分别介绍：

“输入样本”界面：



如图所示，首先需要上传人脸图像、音频数据和姓名标签，可以本地上传，也可以在该界面直接拍照、录音，三处都有数据后，点击“输入样本”，等待“运行结果”显示“输入样本成功”即可。为提高识别的精确度，同一标签可以输入多个人脸、音频数据。待所有样本均输入完毕后，点击“开始训练”按钮即可开始训练，等待“训练结果”显示“训练完成”即可。输入的样本会在训练完成后自动保存，避免每次启动程序都需要重新输入样本，可以直接在检测界面进行检测。

“视频检测：本地视频版”界面：



需要上传视频文件，可以本地上传，也可以在该界面直接拍摄，上传完毕后，点击“开始检测”按钮，等待输出检测结果，会分别输出“到教室的人”和“缺勤的人”，

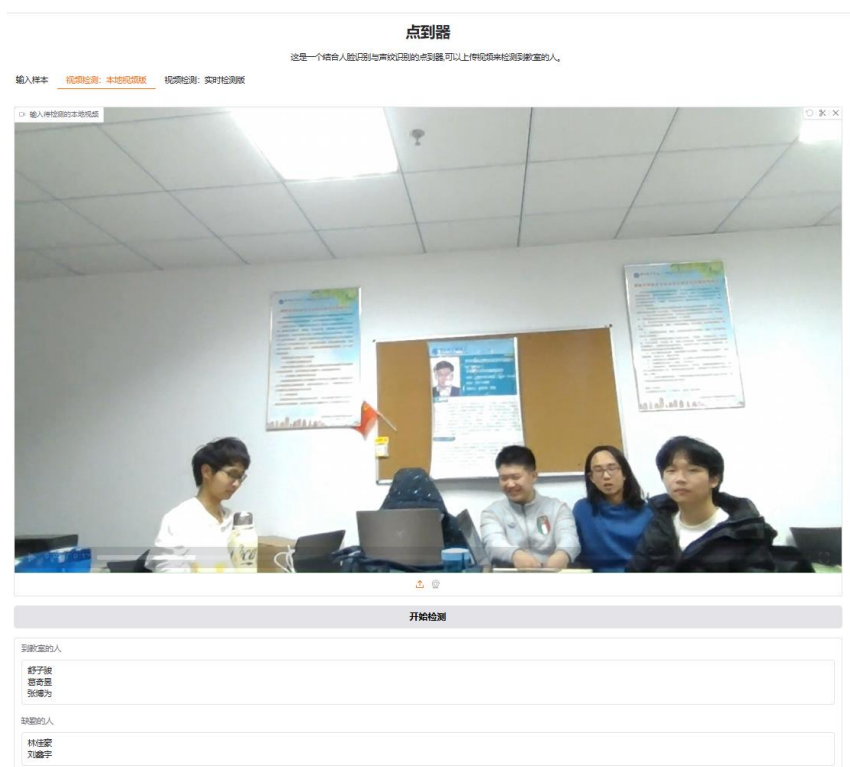
“视频检测：实时检测版”：





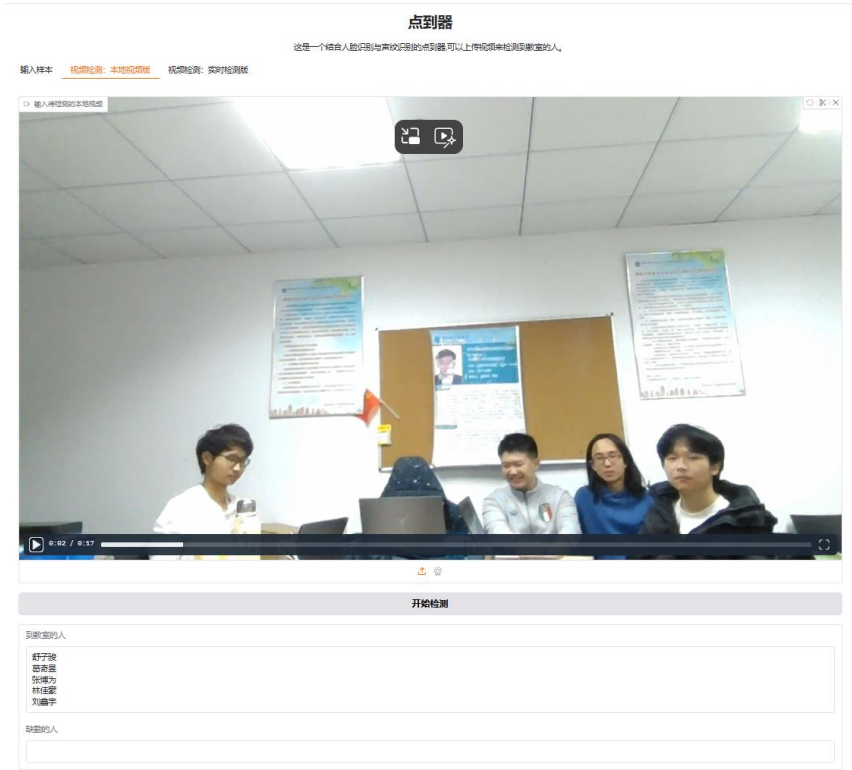
打开音频、摄像头，点击“开始检测”，会根据名单开始播放点名音频，检测有人答到后，会自动播放点下一个人名，同时对输入的数据进行检测，并在“检测结果”输出检测出来的名单。由于检测时分为人脸检测和音频检测，所以只要一方检测出一个标签，最终就会输出这个标签，这样就可以避免因一些原因导致人脸或音频检测错误的情况，达到我们所设想的人脸、音频辅助识别的效果，如下图所示：

图表 1：有人脸部被遮挡（静音）

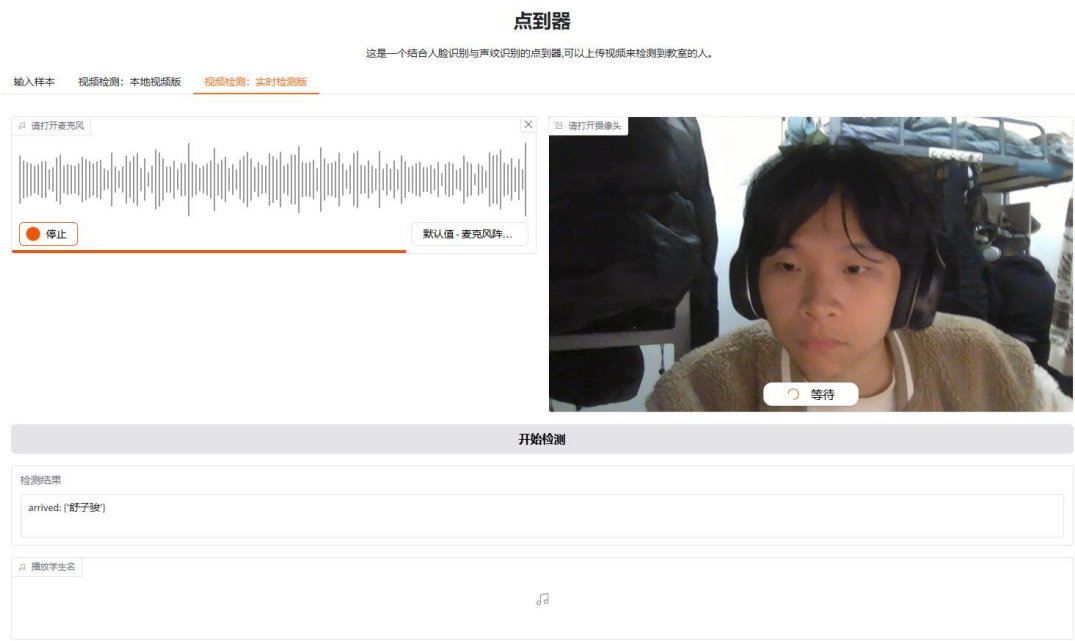


该图为有人脸部被遮挡的静音视频，主要体现人脸识别的结果，结果显示，人脸识别只检测出三个人的人脸，除去脸部被遮挡的人，还有一人可能因为精度原因未被识别出来。

图表 2：有人脸部被遮挡但答到（正常）



该图为正常版本，人脸、语音双重识别，由结果可见，语音识别补齐了人脸识别未检测出来的人，由此可见人脸、语音双重识别的重要性。



如图所示，实时检测出到场人员，检测正确。

#### 四. 研究总结

## 4. 声纹识别

在声纹识别的研究中我们仍有一些问题需要解决：

### ● 视频的答到音频提取

[https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice\\_id.md](https://github.com/SunnyYYLin/vocal-checkin/blob/main/docs/voice_id.md)

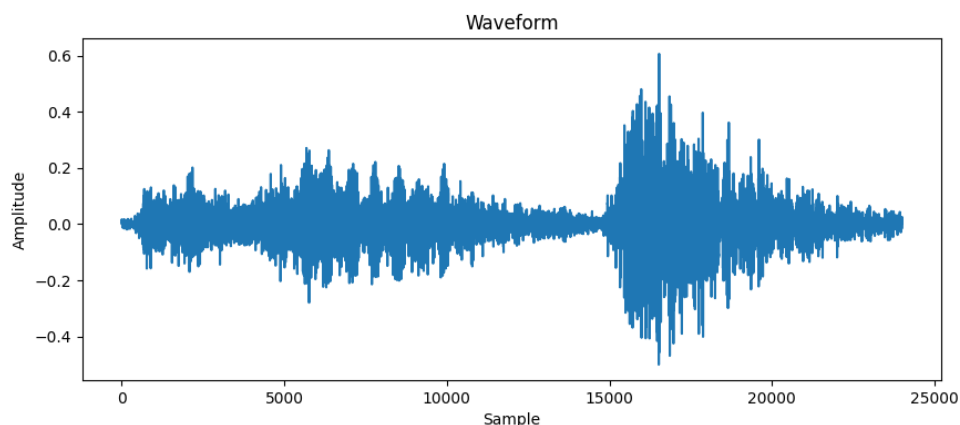
视频的答到音频提取是从多轮对话的连续语音中提取关键词所在音频片段的任务，最靠近关键词唤醒（KWS）任务。但由于数据和精力的限制，多样、大量的答“到”的音频样本难以采集，所以我们采取其他方法实现。由于视频同时存在点到的音频和答到的音频，最开始我们列出的尝试途径有：

- ❖ 利用数据增强训练 KWS 模型，直接用 KWS 模型识别“到”所在片段；
- ❖ 用 VAD 切分音频为每轮问答，或是单独的点到答到，然后使用 KWS 模型识别“到”所在片段；
- ❖ 用 VAD 切分音频为每轮问答，然后使用说话人分离模型分离出点到和答到的片段；
- ❖ 用 ASR 模型识别“到”字，然后获取“到”的时间戳。

由于数据的限制，我们只有小组五个人的答到音频，难以训练一个具有泛化性的 KWS 模型；并且即使使用现有的 TTS 模型生成不同的“到”字音频，答到和“到”音频特点实际上和正常说话中的“到”并不相同，所以我们先搁置了途径 1 和 2。而使用途径 3 和 4 可以利用他人训练好的模型，这值得我们尝试。

对于途径 3，VAD 模型在阈值较高时可以分离出单独的点到答到，但许多答到的音频都没有被识别出；调低阈值后，可以稳定地将音频切分成每段问答。然后我们尝试了 SepFormer 说话人分离模型，但是他的效果并不好，可能是因为一声“到”太短了以至于难以被识别成说话人单独的一句话。所以这条途径也不太可行。

对于途径 4，我们使用了 Whisper，然而可能还是由于同样的原因——一声“到”太短了以至于难以被识别成单独的一句话，单独的“到”甚至不会被 ASR 识别出来，也就无法获取时间戳了。



然后我们画出了途径 3 中 VAD 切分出的每轮问答的波形曲线，发现“到”作为一个激励总是出现在问答的后半段，而点名的音频在中点就已经趋于 0（虽然后面看来这一点的显然的），所以我们又得到了一个途径 5：用 VAD 切分音频为每轮问答，然后直接取后半段的音频。用在教室中录的素材测试，VAD 可以稳定地把音频切分为点到-答到的每一轮，我们以为这个方法就能成功了。然而，当我们重新在研讨室录素材，然后再次用新素材测试，此时 VAD 就完全不稳定了。于是，直到最终，我们都没有得到一个比较可行的“到”的音频提取方案。

### ● 实时的答到音频检测

实时的答到音频检测任务实际上比视频的答到音频提取更加简单，只需要检测是否

有人答到，这就是很典型的 VAD 任务。所以我们当然最开始用 VAD 模型进行检测，我们储存每一轮的音频，直到 VAD 给出这一轮存在至少一段活动音频。但还是遇到了和之前一样的问题——“到”太短了以至于不能准确地激活 VAD 模型。最终我们选择了传统的方法，先得到音频的包络，如果其绝对值超过了某一阈值，就判定为已经答到。

### ● 声纹识别模型性能不足

我们用在研讨室录的视频素材进行测试，发现声纹识别部分的效果很差，于是我们选取了多种情形进行分析。

首先，识别出来的点到音频不是 5 段，这说明可能视频的答到音频提取错误；除此之外，单纯的声纹特征提取模型的性能如何呢？为了排除无关变量，我们利用教室拍摄的视频测试，逐段保存音频，发现每段音频的切分是正常的。然后我们把特征和原型的相似度矩阵打印出来，和人脸特征进行对比。

```
正在识别人脸...
横轴: ['舒子骏', '葛奇昱', '张博为', '林佳豪', '刘鑫宇']
纵轴预测标签: ['刘鑫宇', '舒子骏', '葛奇昱']
相似度分别为: tensor([[-0.0013,  0.0029,  0.0124,  0.0061,  0.9994],
                        [ 0.9987,  0.0149, -0.0103,  0.0187,  0.0085],
                        [ 0.0011,  0.9996, -0.0049,  0.0022,  0.0143]], device='cuda:0')
人脸识别完成!
正在识别声音...
横轴: ['舒子骏', '葛奇昱', '张博为', '林佳豪', '刘鑫宇']
纵轴预测标签: ['刘鑫宇', '葛奇昱', '葛奇昱', '刘鑫宇', '刘鑫宇']
相似度分别为: tensor([[-0.0045,  0.3766,  0.6092,  0.2487,  0.6234],
                        [ 0.1097,  0.8255,  0.1839,  0.1969,  0.3610],
                        [-0.0081,  0.8439,  0.2621,  0.1129,  0.3819],
                        [-0.0116,  0.2529,  0.0366,  0.3323,  0.8532],
                        [ 0.0543,  0.0657,  0.4919,  0.0508,  0.8592]], device='cuda:0')
声音识别完成!
```

发现人脸会被遮挡，导致识别不全，但是只要识别到了，其特征和原型相似度就极高，都是 99.5% 以上；反观声纹特征，在教室场景下被正确提取并分割了，但特征和原型的匹配度不高，正确或者不正确的情况下都会有 80%+。

随后，我们又测试了单独录音，然后查看相似度。

```
横轴: ['舒子骏', '葛奇昱', '张博为', '林佳豪', '刘鑫宇']
纵轴预测标签: 林佳豪
相似度分别为: tensor([0.0287, 0.0122, 0.0790, 0.9286, 0.3486],
device='cuda:0')
```

然后我们发现，单独对着电脑录音然后检测，就大概率能识别正确，且相似度可以达到 90%+。据此，我们猜测，声纹特征提取模型的性能不佳可能是因为测试环境数据和录入标签数据差异比较大，尤其是教室等空间较大的环境，可能会加入混响。之后可以尝试利用去混响模型对数据进行预处理，但是我们也担心这会影响到声纹的辨认。

## 5. 人脸识别

总体来说由于前期进行了比较详细的调查，基础功能实现起来并没有遇到太多困难，但是有很多实际运行起来后遇到的小问题：原方法返回值类型不统一（当未检测到人脸时会直接返回 None 而非空列表）导致与其他模块无法顺利结合；所需的 torch 版本又老又严格，险些无法与其他模块统一等等。这些都提醒我在合作过程中，要提前与他人



进行详细的沟通，否则在更大的项目中可能会遇到更多类似的小问题，最终导致无法运行。

而更大的问题同样存在：由于我的方法中使用的检测和提取方法运行都较慢，因此主程序不得不采用从每秒中只提取其中一两帧传给我进行检测，而这实际上与我的连续帧检测逻辑上不太相容：跳跃帧数过多的情况下人脸很有可能会在不同区域间跳跃，导致正确人脸无法被检测。因此我只能将阈值设置的很低，所以实际工作中起到的效果并不是很大。

这也警示我在实现新功能时，不仅要考虑这个问题本身，还要更多的考虑大的框架以及和其他模块的配合问题，才能保证实现的新功能对于整个系统是有效的。

未来如果要进一步地优化该工具，我认为可能要考虑轻量化以及过滤功能中人脸连续出现的检测区域设置的优化问题。因为目前整个系统在人脸检测模块上耗费的时间最多，进而导致了检测数据要十几倍地减少，并因此降低了检测精度和过滤功能。同时在测试过程中也发现了即使是目前只有五个人的情况，也有可能出现多个人脸检测区域大量重合的情况，面对更复杂的情况时这种方法可能会存在更多实际问题，这方面还需要更多巧思来进行优化。

## 6. 数据存储、孪生网络训练与识别

从简单到复杂、从传统机器学习到深度学习的转变过程，每一次改进都基于前一阶段方法的优缺点总结，以及对特征分布、任务需求的更深入理解。孪生网络作为最终方案，不仅解决了传统方法的瓶颈，还为特征向量识别提供了一种通用且高效的解决思路。尽管这一网络较为简单、易于实现，但是对于从未进行过相关内容编程的我来说，确实是一次非常宝贵的学习机会。过程中所遇到的各种困难，例如模型的训练与应用、数据结构的调整、数据的存储及调用等都在一次又一次的查找中得到解决方法。

从代码的角度来说，database 算不上一个有什么含金量的部分，虽然自动阈值非常有亮点，但是孪生网络和损失函数都是选用简单的而非有深度的版本。然而 database 算得上是麻雀虽小五脏俱全，未来经过系统性的学习后进行更贴合实际情况的修改，想必也能可以广泛应用。

总而言之，经过整个小组的通力协作与集思广益，我们的智能考勤系统已经能够基本实现本地与实时的考勤功能。但是由于时间与经费的限制，我们的研究仍有进一步的拓展空间。首先，我们可以进一步收集点到数据，并训练语音“到”的关键词唤醒模型，提高程序的自动化程度。同时，可以尝试增量学习，同一样本只用采样一次，大大降低数据库的收集成本。此外，我们可以尝试将模型轻量化处理，使之可以集成到可移动设备上。

## 五. 附件

本项目已经开源在 GitHub:

<https://github.com/SunnyYYLin/video-checkin>

## 六. 贡献

林佳豪、张博为、刘鑫宇分别完成了声音处理模块、人脸处理模块、数据库模块，葛奇昱主要完成了 UI，舒子骏主要完成了主程序部分。林佳豪参与了数据库模块和主程序的 debug 和增强。林佳豪、张博为、刘鑫宇分别写了自己模块部分相关的报告，舒子骏介绍了主程序的流程并完成了效果演示部分，由葛奇昱和整合并完成了头尾。

## 七. 参考

1. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. arXiv preprint arXiv:1503.03832.
2. Desplanques, B., Thienpondt, J., & Demuynck, K. (2020). Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. *arXiv preprint arXiv:2005.07143*.
3. snakers4. (2024). *Silero VAD: Silero Voice Activity Detector*. GitHub. <https://github.com/snakers4/silero-vad>
4. SpeechBrain. (2021). SpeechBrain. GitHub. <https://github.com/speechbrain/speechbrain>