Wav2lip模型实现数字人唇语同步实验报告

马俊晨1

（1.北京理工大学，北京 100083 ）

摘 要：现阶段人脸语音合成仍然是大模型研究方向的热门领域之一，本文将以github上的Wav2lip项目为基础，详细说明如何对人脸语音合成进行复现，以及选用模型的方式和原因，最后本文会说明如何对生成出来的视频进行评估，包括从syncnet和pytorch\_FID两种方法对生成出来的视频进行定量评估和定性评估。

关键词： 语音合成，大模型，Wav2lip，FID，LSE-C, LSE-D

Report on the Experiment of Lip Synchronization for Digital Humans Using the Wav2lip Model

Ma Jun-chen1

1. Beijing Institute of Technology, Beijing 100083)

Abstract: At present, face-to-speech synthesis remains one of the hottest research areas in the field of large models. This article will be based on the Wav2lip project on GitHub, and will provide a detailed explanation on how to reproduce face-to-speech synthesis, as well as the methods and reasons for selecting models. Finally, this article will explain how to evaluate the generated videos, including quantitative and qualitative evaluations of the generated videos using both syncnet and pytorch\_FID methods.

Keywords: Speech synthesis, Large model, Wav2lip, FID, LSE-C, LSE-D

1. 引言

人脸语音合成一直作为大模型研究领域的热门话题，当我们拥有一张人脸图片或者是一段人像视频，我们应该怎样把这些与我们想要的音频拼接在一起，达到让一个人说我们想要的话呢，接下来本文就会通过对Wav2lip模型的使用来复现这一操作，将我们提供的测试集视频中每个视频的第一帧作为我们的人脸输入，将每个视频的音频分离出来作为我们的音频输入，最后会对我们生成的视频进行两种评估，一种是基于syncnet方法的无监督评估，一种是基于pytorchd\_fid库函数自带的图像对比进行比较评估。

1. 模型简介
   1. Wav2lip

该模型是在Wav2lip项目中提供的第一个预训练模型，它通过在LRS2数据集上进行训练。Wav2Lip使用了一种基于深度学习的生成对抗网络（GAN）架构。该模型可以解决面部动画与音频之间的同步问题。具体来说，它能够根据输入的音频信号生成与之匹配的面部表情和口型，使得视频中的人脸能够与音频的语音内容自然同步。这一技术在许多应用场景中具有重要意义，例如视频配音、虚拟角色的表情生成和电影后期制作等。具体而言，模型采用了一个编码器-解码器结构，首先提取音频信息并将其编码，然后生成与之相匹配的面部图像。模型训练过程中使用了大量的音频-视频配对数据，通过对抗训练使得生成的口型与真实视频中的口型尽可能一致。Wav2Lip在各种基准测试中表现出色，能够生成高质量的面部动画。其生成的口型与音频的同步性非常高，能够有效地捕捉细微的面部表情变化。此外，该模型在不同语音和说话者的情况下都表现良好，显示出较强的泛化能力。

* 1. Wav2lip\_gan

该模型是项目中提供的另一个预训练模型，Wav2Lip\_GAN是Wav2Lip模型的扩展，旨在进一步提升生成口型的质量和真实感。Wav2Lip\_GAN在Wav2Lip的基础上引入了改进的生成对抗网络机制。它结合了条件GAN的思想，通过引入额外的损失函数来增强生成图像的细节，并提高模型对不同面部表情和动态语音的适应能力此外，Wav2Lip\_GAN在训练过程中使用了更丰富的训练数据集，包含了多种口型和表情的样本，以提升模型的表现。Wav2Lip\_GAN在生成质量和真实感上相较于原始Wav2Lip模型有显著提升。与Wav2Lip相比，Wav2Lip\_GAN更注重生成的面部动画的细节和真实感，尤其是在动态场景中，确保生成的视频在视觉上更加自然和流畅，但是作为代价，该模型的口型同步略差。

1. 实验困难及解决方案
   1. 虚拟环境配置

首先是虚拟环境的配置，在windows运行时，我们首先要使用conda创建虚拟环境，这里需要选择python=3.8版本，接着我们需要安装wav2lip项目中需要的库以及包，当我直接按照项目中提供的requirement.txt进行下载时，出现了报错，所以这里我们对一些包需要选择正确的版本，例如librosa需要选择0.8.0版本，项目中提供的numpy版本过低，但是版本过高的话也会导致后续代码无法运行，所以这里需要选择1.23.3版本，这里要注意的是不管是在windows还是在docker中都需要cuda=11.8版本，不然在评估时会出现报错。同时，在评估方法中我们用到了依赖于syncnet网络的LSE-C和LSE-D分数评估方法和pytorch\_fid评估方法，我们需要额外下载需要的库。

* 1. 实验结果生成

在结果生成中遇到的问题分为生成视频结果和评估结果两方面。首先是生成视频中遇到的问题，在该模型中，我们可以选择图片或者视频作为模型的人脸输入，当我们使用视屏作为人脸输入时，如果用cpu运行模型，运行程序时间会比较长，并且对于wav2lip\_gan模型，可能会出现程序卡死的情况，此时有两种方法，一种是通过修改inference.py文件中的bitch\_size，我们需要限定块的大小（最小值为1），另一种方法就是我使用的，在windows下我们可以通过conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia命令来安装cuda11.8，此时windows下就可以自动调用gpu来运行模型，而在docker中，我们需要提前在docker run阶段设置好gpu运行。另一方面是关于评估结果生成的问题，我在使用syncnet的无监督评测方法时发现需要自行下载模型，这个在原项目中我没有找到，所以从网上下载了预训练的模型。使用pytorch\_fid方法，本质上是将两个文件夹的图像进行比对生成FID指数，所以这里需要我们先自行 编写代码将生成视频和输入视频按照帧提取为图像，再通过python -m pytorch\_fid比较两个文件夹，这一步最大的困难是windows和docker下运行起来不相同，在docker运行该命令时，会报错显示共享内存不够用，我首先尝试用--shm-size=3g扩大docker的共享内存为3g，但依旧报错，在上网查阅资料后，在命令后添加--num-workers 0 --device cuda:0，程序正常运行。

* 1. docker封装

这一部分遇到的问题是我在本次实验中遇到问题最多的一部分，因为在此之前我并没有使用过docker，所以首先遇到的问题就是如何使用docker，在docker中拉取镜像时，因为网络问题会显示无法拉取，所以此时需要我们通过国内的镜像网站来进行拉取，同时还有在工作区域内挂载我们要需要用到的目录，为了确保模型正常运行，需要在docker run阶段输入--gpus all和--shm-size=3g设置共享内存和gpu运行。

1. 模型的定性及定量评价结果及可能改进方法

4.1模型的定量评估

在生成唇语对应视频时我们选择使用wav2lip\_gan模型，我们选取了包含八个视频的测试集（评价数据集[下载链接](https://drive.google.com/drive/folders/1FwQoBd1ZrBJMrJE3ZzlNhK8xAe1OYGjX)）作为我们的测试数据，我们选择每个测试视频的第一帧图像作为模型需要的人脸数据，把每个视频的音频分离为

WAV文件作为模型输入的音频文件，最终生成为我们需要的语音合成视频，接着我们从LSE-C、LSE-D分数和FID指数两个方面对这些生成视频进行定量评估。

4.1.1 LSE-C以及LSE-D分数评估

运用syncnet项目文件对生成视频进行评测

|  |  |  |
| --- | --- | --- |
| 生成视频（用于生成视频的测试视频） | LSE-C | LSE-D |
| result1.mp4(Jae-in.mp4) | 6.2492943 | 7.7773724 |
| result2.mp4(Lieu.mp4) | 5.8220344 | 10.318459 |
| result3.mp4(Macron.mp4) | 5.801061 | 8.821535 |
| result4.mp4(May.mp4) | 5.4944816 | 10.029444 |
| result5.mp4(Obama.mp4) | 5.498189 | 9.814372 |
| result6.mp4(Obama1.mp4) | 6.336245 | 9.415861 |
| result7.mp4(Obama2.mp4) | 5.6072683 | 8.995091 |
| result8.mp4(Shaheen.mp4) | 5.914622 | 10.038904 |

基于LSE-C和LSE-D分数的生成视频评估结果

LSE-C测量的是在时间上连续的音频和视频之间的同步误差。它评估的是在整个视频播放过程中嘴形与音频的匹配程度，从上表我们可以得知该模型的生成视频的LSE-C指数基本在5-7之间。LSE-D则测量特定时间点（通常是关键帧或特定的发音时刻）下的音视频同步误差。它专注于离散的时间点，评估这些时间点上嘴唇动作与音频的匹配。从上表我们可以得知该模型的生成视频的LSE-D指数基本在7-10之间。

4.2.2 FID指数评估

运用pytorch\_fid库自带命令行对生成视频进行评测

|  |  |
| --- | --- |
| 生成视频（用于生成视频的测试视频） | FID指数 |
| result1.mp4(Jae-in.mp4) | FID: 20.354803087631055 |
| result2.mp4(Lieu.mp4) | FID: 44.9185183060483 |
| result3.mp4(Macron.mp4) | FID: 14.43842736648849 |
| result4.mp4(May.mp4) | FID: 58.13635265447473 |
| result5.mp4(Obama.mp4) | FID: 35.58621571873616 |
| result6.mp4(Obama1.mp4) | FID: 36.02404854340562 |
| result7.mp4(Obama2.mp4) | FID: 21.52632009852146 |
| result8.mp4(Shaheen.mp4) | FID: 32.18086089809941 |

基于FID指数的生成视频评估结果

FID是评估生成图像质量的常用指标，数值越低，表示生成的图像与真实图像之间的相似性越高。从表中我们可以得知结果1和结果7的表现相对较好，而结果2、结果5、结果6和结果8 的 FID 值显示出不同程度的生成质量下降。这里的原因可能是由于我们将图片作为人脸输入，这就导致在生成视频中任务除了嘴唇及周围部分外是静态的，对于人物表情变化多，活动幅度大的视频，生成效果就会变差。

4.2 模型的定性评估

对于模型的定性评估，我会从以下几个方面进行：视觉一致性，面部表情和细节以及多样性适应能力。

4.2.1 视觉一致性

在这一部分，通过直接观察的方法来对生成视频和测试视频，我们发现在输入的人脸图像为图片时，整个视频会更加稳定，唇型匹配度会更高，但是会观察到人脸明显的僵硬，与原视频的视觉一致性不高，而当我们将输入的人脸图像改为视频时，我们发现生成视频明显与原视频之间更接近，但是视频有时会出现失真的问题。而对于模型选择来说，使用wav2lip模型会获得唇型匹配度更高的生成视频，而使用wav2lip\_gan模型会获得更加流畅和高画质的视频。

4.2.2 面部表情和细节

在使用wav2lip模型时，生成的视频面部表情细节明显不如wav2lip\_gan模型下生成的视频。而使用图片作为人脸输入的话，面部几乎无表情，所以如果我们需要更好的面部表情和细节，就必须选用视频作为我们的人脸输入。

4.2.3多样性适应能力

在该能力下主要测试的在不同语言和口音的音频中，模型的表现是否一致、是否能正确处理不同的发音特征以及在各种视频背景和光照条件下，模型的表现是否稳定。经过我的测试发现背景的不同基本上不会改变模型的输出质量，但是在画质过高或者是视频中没有清晰的正面人脸时，生成的视频会经常在唇部失真。

4.3模型存在问题

由于本实验中使用的是预训练模型，所以我们仅从运行和结果方面来观察模型现有问题，首先是生成视频的画质问题，对于本身高画质的输入视频，在经过模型处理后画质会下降并出现失真情况，模型应对多样性的能力并不强，对于视频中的人物转头，抖动动作的处理并不出色，表现形式为，当视频中的人物转头时，可能出现说话的嘴在人物脖子上这种情况。

1. **总结心得**

在本次实验中我选择了wav2lip项目来进行实验，选择这项项目的主要原因还是自己对自己的能力并不是很自信，所以希望通过这个简单的项目来对语音识别这一领域有自己的了解，一开始觉得自己选择了所有项目中最简单的，会不会在很短的时间内就把这个任务完成，但是在真正做的时候才发现自己好像自己还是高估了自己的能力，就算模型都是别人预训练好的，配置虚拟环境和保证兼容性也废了半天时间，老实说通过对该项目的复现，我觉得最大的收获反而是安装并使用了docker，在此之前我并没有使用过这个软件，第一遍项目是在windows下安装虚拟环境跑的，在运行完毕后都觉得任务已经完成了，但是没想到分装到docker这步对我来说才是最大的阻碍，只能去查资料和问ai，然后把需要使用到的命令写在一起，直到现在我还是对于docker还是个小白，总的来说，我个人还是对我这次实验挺满意的。