

## 21 | DID和PaddleGAN：表情生动的数字人播报员

2023-04-24 徐文浩 来自北京

《AI大模型之美》



你好，我是徐文浩。

上一讲里，我们已经学会了通过 AI 来进行语音合成。有了语音识别、ChatGPT，再加上这个语音合成，我们就可以做一个能和我们语音聊天的机器人了。不过光有声音还不够，我们还希望这个声音可以是某一个特定的人的声音。就好像在电影《Her》里面那样，AI 因为用了影星斯嘉丽·约翰逊的配音，也吸引到不少观众。最后，光有声音还不够，我们还希望能够有视觉上的效果，最好能够模拟自己真的在镜头面前侃侃而谈的样子。


这些需求结合在一起，就是最近市面上很火的“数字人”，也是我们这一讲要学习的内容。当然，在这么短的时间里，我们做出来的数字人的效果肯定比不上商业公司的方案。不过作为概念演示也完全够用了。

### 制作一个语音聊天机器人

#### 从文本 ChatBot 起步

我们先从最简单的文本 ChatBot 起步，先来做一个和 [第 6 讲](#) 一样的文本聊天机器人。对应的代码逻辑和第 6 讲的 ChatGPT 应用基本一样，整个的 UI 界面也还是使用 Gradio 来创建。

唯一的区别在于，我们把原先自己封装的 Conversation 类换成了 Langchain 的 ConversationChain 来实现，并且使用了 SummaryBufferMemory。这样，我们就不需要强行设定只保留过去几轮对话了。

 复制代码

```
1 import openai, os
2 import gradio as gr
3 from langchain import OpenAI
4 from langchain.chains import ConversationChain
5 from langchain.memory import ConversationSummaryBufferMemory
6 from langchain.chat_models import ChatOpenAI
7
8 openai.api_key = os.environ["OPENAI_API_KEY"]
9
10 memory = ConversationSummaryBufferMemory(llm=ChatOpenAI(), max_token_limit=2048)
11 conversation = ConversationChain(
12     llm=OpenAI(max_tokens=2048, temperature=0.5),
13     memory=memory,
14 )
15
16 def predict(input, history=[]):
17     history.append(input)
18     response = conversation.predict(input=input)
19     history.append(response)
20     responses = [(u,b) for u,b in zip(history[::2], history[1::2])]
21     return responses, history
22
23 with gr.Blocks(css="#chatbot{height:800px} .overflow-y-auto{height:800px}") as de
24     chatbot = gr.Chatbot(elem_id="chatbot")
25     state = gr.State([])
26
27     with gr.Row():
28         txt = gr.Textbox(show_label=False, placeholder="Enter text and press ente
29
30         txt.submit(predict, [txt, state], [chatbot, state])
31
32 demo.launch()
```


对应界面：



## 增加语音输入功能

接着，我们来给这个聊天机器人加上语音输入的功能，Gradio 自带 Audio 模块，所以要做到这一点也不难。

1. 首先，我们在 Gradio 的界面代码里面增加一个 Audio 组件。这个组件可以录制你的麦克风的声。

 复制代码

```
1 with gr.Row():
```

```
2         txt = gr.Textbox(show_label=False, placeholder="Enter text and press ente
```

2. 然后，我们封装了一个 transcribe 方法，通过调用 OpenAI 的 Whisper API 就能够完成语音识别。这里有一点需要注意，OpenAI 的 Whisper API 有点笨，它是根据文件名的后缀来判断是否是它支持的文件格式的。而 Gradio 的 Audio 组件录制出来的 WAV 文件没有后缀，所以我们要在这里做个文件重命名的工作。

 复制代码

```
1 def transcribe(audio):
2     os.rename(audio, audio + '.wav')
3     audio_file = open(audio + '.wav', "rb")
4     transcript = openai.Audio.transcribe("whisper-1", audio_file)
5     return transcript['text']
```

3. 接着，我们就要把麦克风录好的声音自动发送给语音识别，然后再提交给原先基于文本聊天的机器人就好了。

 复制代码

```
1     audio.change(process_audio, [audio, state], [chatbot, state])
```

我们先在 Audio 的 change 事件里，定义了触发 process\_audio 的函数。这样，一旦麦克风的声音录制下来，就会直接触发聊天对话，不需要再单独手工提交一次内容。

 复制代码

```
1 def process_audio(audio, history=[]):
2     text = transcribe(audio)
3     return predict(text, history)
```

然后在 process\_audio 函数里，我们先是转录对应的文本，再调用文本聊天机器人的 predict 函数，触发对话。

修改后的完整代码在下面，你可以在本地运行，体验一下。

```

1 import openai, os
2 import gradio as gr
3 import azure.cognitiveservices.speech as speechsdk
4 from langchain import OpenAI
5 from langchain.chains import ConversationChain
6 from langchain.memory import ConversationSummaryBufferMemory
7 from langchain.chat_models import ChatOpenAI
8
9 openai.api_key = os.environ["OPENAI_API_KEY"]
10
11 memory = ConversationSummaryBufferMemory(llm=ChatOpenAI(), max_token_limit=2048)
12 conversation = ConversationChain(
13     llm=OpenAI(max_tokens=2048, temperature=0.5),
14     memory=memory,
15 )
16
17 def predict(input, history=[]):
18     history.append(input)
19     response = conversation.predict(input=input)
20     history.append(response)
21     responses = [(u,b) for u,b in zip(history[::2], history[1::2])]
22     return responses, history
23
24 def transcribe(audio):
25     os.rename(audio, audio + '.wav')
26     audio_file = open(audio + '.wav', "rb")
27     transcript = openai.Audio.transcribe("whisper-1", audio_file)
28     return transcript['text']
29
30 def process_audio(audio, history=[]):
31     text = transcribe(audio)
32     return predict(text, history)
33
34 with gr.Blocks(css="#chatbot{height:350px} .overflow-y-auto{height:500px}") as de
35     chatbot = gr.Chatbot(elem_id="chatbot")
36     state = gr.State([])
37
38     with gr.Row():
39         txt = gr.Textbox(show_label=False, placeholder="Enter text and press ente
40
41     with gr.Row():
42         audio = gr.Audio(source="microphone", type="filepath")
43
44     txt.submit(predict, [txt, state], [chatbot, state])
45     audio.change(process_audio, [audio, state], [chatbot, state])
46
47 demo.launch()

```

对应界面：




想要录新的一句话，点击红色方框内的X，重新进入录音界面

## 增加语音回复功能

在能够接收语音输入之后，我们要做的就是让 AI 也能够用语音来回答我们的问题。而这个功能，通过 [上一讲](#) 我们介绍过的 Azure 的语音合成功能就能实现。我们只需要封装一个函


数，来实现语音合成与播放的功能，然后在 predict 函数里面，拿到 ChatGPT 返回的回答之后调用一下这个函数就好了。

## 1. 封装一个函数进行语音合成与播放。

 复制代码


```
1
2 speech_config = speechsdk.SpeechConfig(subscription=os.environ.get('AZURE_SPEECH_
3 audio_config = speechsdk.audio.AudioOutputConfig(use_default_speaker=True)
4
5 # The language of the voice that speaks.
6 speech_config.speech_synthesis_language='zh-CN'
7 speech_config.speech_synthesis_voice_name='zh-CN-XiaohanNeural'
8
9 speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config, aud
10
11 def play_voice(text):
12     speech_synthesizer.speak_text_async(text)
13
```

## 2. 在拿到 ChatGPT 的返回结果之后调用一下这个函数。

 复制代码

```
1 def predict(input, history=[]):
2     history.append(input)
3     response = conversation.predict(input=input)
4     history.append(response)
5     play_voice(response)
6     responses = [(u,b) for u,b in zip(history[::2], history[1::2])]
7     return responses, history
```

完整的语音对话的 Demo 代码我一并放在了下面，你可以像 [🔗 第 6 讲](#) 里我们介绍过的那样，直接部署到 Gradio 里面体验一下分享出去。

 复制代码

```
1 import openai, os
2 import gradio as gr
3 import azure.cognitiveservices.speech as speechsdk
```

```

4 from langchain import OpenAI
5 from langchain.chains import ConversationChain
6 from langchain.memory import ConversationSummaryBufferMemory
7 from langchain.chat_models import ChatOpenAI
8
9 openai.api_key = os.environ["OPENAI_API_KEY"]
10
11 memory = ConversationSummaryBufferMemory(llm=ChatOpenAI(), max_token_limit=2048)
12 conversation = ConversationChain(
13     llm=OpenAI(max_tokens=2048, temperature=0.5),
14     memory=memory,
15 )
16
17 speech_config = speechsdk.SpeechConfig(subscription=os.environ.get('AZURE_SPEECH_
18 audio_config = speechsdk.audio.AudioOutputConfig(use_default_speaker=True)
19
20 # The language of the voice that speaks.
21 speech_config.speech_synthesis_language='zh-CN'
22 speech_config.speech_synthesis_voice_name='zh-CN-XiaohanNeural'
23
24 speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config, aud
25
26 def play_voice(text):
27     speech_synthesizer.speak_text_async(text)
28
29 def predict(input, history=[]):
30     history.append(input)
31     response = conversation.predict(input=input)
32     history.append(response)
33     play_voice(response)
34     responses = [(u,b) for u,b in zip(history[::2], history[1::2])]
35     return responses, history
36
37 def transcribe(audio):
38     os.rename(audio, audio + '.wav')
39     audio_file = open(audio + '.wav', "rb")
40     transcript = openai.Audio.transcribe("whisper-1", audio_file)
41     return transcript['text']
42
43 def process_audio(audio, history=[]):
44     text = transcribe(audio)
45     return predict(text, history)
46
47 with gr.Blocks(css="#chatbot{height:800px} .overflow-y-auto{height:800px}") as de
48     chatbot = gr.Chatbot(elem_id="chatbot")
49     state = gr.State([])
50
51     with gr.Row():
52         txt = gr.Textbox(show_label=False, placeholder="Enter text and press ente

```



```
53
54     with gr.Row():
55         audio = gr.Audio(source="microphone", type="filepath")
56
57     txt.submit(predict, [txt, state], [chatbot, state])
58     audio.change(process_audio, [audio, state], [chatbot, state])
59
60 demo.launch()
```

## 用 D-ID 给语音对口型

这里我们设计的聊天机器人不仅能够完全听懂我们说的话，还能通过语音来对话，这的确是一件很酷的事情。而且这里我们算上空行，也只用了 60 行代码。不过，我们并不会止步于此。接下来，我们还要为这个聊天机器人配上视频画面和口型。

现在，国内外已经有一些公司开始提供基于 AI 生成能对上口型的“数字人”的业务了。这里，我们就来试试目前用户比较多的 [D-ID](#) 提供的 API，毕竟它直接为所有开发者提供了开放平台，并且还有 5 分钟的免费额度。

### 通过 D-ID 生成视频

首先，你要去 [d-id.com](#) 注册一个账号。别紧张，[d-id.com](#) 有邮箱就能注册账号，不像 ChatGPT 那么麻烦，并且 D-ID 送给注册用户 20 次调用 API 的机会，我们可以好好利用这些免费额度。

注册好账号以后，你就可以去访问自己的 [Account Setting](#) 页面生成一个 API\_KEY 了。


### Profile Details



Full name

Email

### API Key

Create videos at scale, learn more by reading our [API documentation](#) 



No API key to show yet



A trial API key is restricted and has a lower priority. We'll upgrade it automatically once you subscribe.

GENERATE NEW KEY

之后，你可以查看一下 D-ID 的文档，里面不仅有 API 的使用说明，还有一个类似 Playground 的界面，你可以设置参数，并且可以测试调用 API。

**CREATIVEREALITY™**

v4.2.0 <> API Reference Discussions

JUMP TO 36/

WELCOME  
Getting Started 🚀

AUTHENTICATION  
Basic Authentication

TALKS  
Talks  
Overview 📄  
Create a talk POST  
Get a specific talk GET  
Get talks GET  
Delete a specific talk DELETE

Clips  
Animations  
Streams

RESOURCES ENDPOINTS  
Images  
Audios  
Credits  
Settings

TEXT TO SPEECH PROVIDERS  
Microsoft Azure Voices  
Amazon Polly Voices

ERRORS  
Asynchronous Errors

Powered by readme

## Create a talk

POST <https://api.d-id.com/talks>

Create a talk

YOUR REQUEST HISTORY

2 Calls 7 Days

TIME	STATUS	PATH	USER AGENT
4/11/2023 11:20 PM	201	/talks	API Expl...
4/11/2023 11:19 PM	400	/talks	API Expl...

Page 1

**Input**  
Photo URL + Text or Audio file URL

**Output**  
Video URL

Photo Text / Audio Video

**BODY PARAMS**

**source\_url** string required   
The URL of the source image to be animated by the driver video, or a selection from the list of provided studio actors.

**driver\_url** string   
The URL of the driver video to drive the talk, or a selection from the list or provided drivers  
If not provided a driver video will be selected for you from the predefined drivers bank

**LANGUAGE**  
Shell Node Ruby PHP Python

**AUTHENTICATION**  
Bearer

**INSTALLATION**  
`$ python -m pip install requests`

**REQUEST** **EXAMPLES**

```
1 import requests
2
3 url = "https://api.d-id.com/talks"
4
5 headers = {
6     "accept": "application/json",
7     "content-type": "application/json",
8     "authorization": "Bearer "
9 }
10
11 response = requests.post(url, headers=headers)
12
13 print(response.text)
```

**RESPONSE** **EXAMPLES**

Click **Try It!** to start a request and see the response here!  
Or choose an example:

application/json

201 - Example 1 400 - Example 1  
401 - Example 1 402 - Example 1  
403 - Example 1 451 - Example 1

我们设置一下对应的 API KEY 并且确保安装了 requests 这个专门用来写 HTTP 请求的 Python 包，就可以测试一下这个代码的效果了。

安装 requests 包：

```
1 pip install requests
```

复制代码

设置 DID\_API\_KEY 的环境变量：

```
1 export DID_API_KEY=YOUR_DID_API_KEY
```


我们可以先调用 D-ID 的 **Create A Talk** 接口，来创建一段小视频。只需要输入两个东西：一个是我们希望这个视频念出来的文本信息 input，另一个就是一个清晰的正面头像照片。

在下面的代码里面可以看到，这其实就是一个简单的 HTTP 请求，并且文本转换成语音的过程，其实调用的也是 Azure 的语音合成功能。

```
1 import requests
2 import os
3
4 def generate_talk(input, avatar_url,
5                  voice_type = "microsoft",
6                  voice_id = "zh-CN-XiaomoNeural",
7                  api_key = os.environ.get('DID_API_KEY')):
8     url = "https://api.d-id.com/talks"
9     payload = {
10         "script": {
11             "type": "text",
12             "provider": {
13                 "type": voice_type,
14                 "voice_id": voice_id
15             },
16             "ssml": "false",
17             "input": input
18         },
19         "config": {
20             "fluent": "false",
21             "pad_audio": "0.0"
22         },
23         "source_url": avatar_url
24     }
25     headers = {
26         "accept": "application/json",
27         "content-type": "application/json",
28         "authorization": "Basic " + api_key
29     }
30
31     response = requests.post(url, json=payload, headers=headers)
32     return response.json()
33
```

```
34 avatar_url = "https://cdn.discordapp.com/attachments/1065596492796153856/10956174
35 text = "今天天气真不错呀。"
36
37 response = generate_talk(input=text, avatar_url=avatar_url)
38 print(response)
```


输出结果：

 复制代码

```
1 {'id': 'tlk_Nk90fTGu_ZvLztD3HHC4b', 'created_at': '2023-04-12T03:07:38.593Z', 'cr
```


这段代码运行成功之后，返回的结果是一个 JSON。JSON 里面有一个对应视频的 id，我们可以通过这个 id 用 Get A Talk 的 API 拿到我们刚刚生成的口播视频，然后在 Notebook 里面播放。

获取生成的 Talk 视频：

 复制代码

```
1 def get_a_talk(id, api_key = os.environ.get('DID_API_KEY')):
2     url = "https://api.d-id.com/talks/" + id
3     headers = {
4         "accept": "application/json",
5         "authorization": "Basic "+api_key
6     }
7     response = requests.get(url, headers=headers)
8     return response.json()
9
10 talk = get_a_talk(response['id'])
11 print(talk)
```

输出结果：

 复制代码

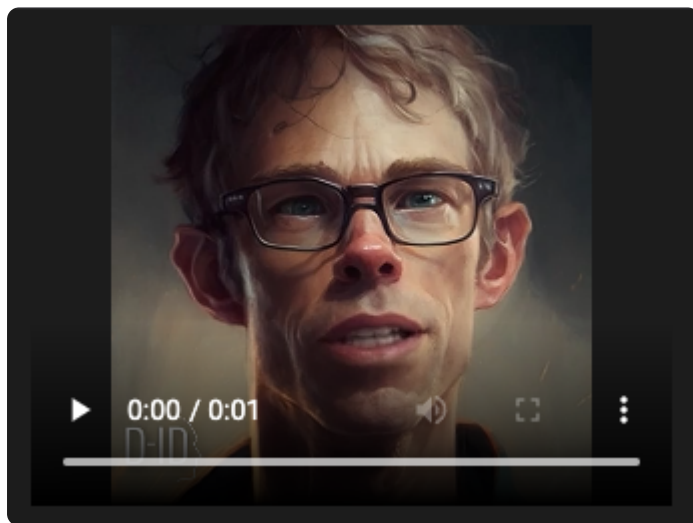
```
1 {'metadata': {'driver_url': 'bank://lively/driver-03/original', 'mouth_open': Fal
```

将对应的视频展示播放出来：

📄 复制代码

```
1 from IPython.display import display, HTML
2 def play_mp4_video(url):
3     video_tag = f"""
4     <video width="640" height="480" controls>
5         <source src="{url}" type="video/mp4">
6     Your browser does not support the video tag.
7     </video>
8     """
9     return HTML(video_tag)
10 result_url = talk['result_url'])
11 play_mp4_video(result_url)
```

输出展示：




在这里，我用 Midjourney 生成了一张 ID Software 的创始人——大神约翰卡马克的头像。然后让 D-ID 给这个头像生成对应的对口型的视频，看到心目中的技术偶像开口说话还是非常让人震撼的。

## 将视频嵌入到 Gradio 应用中

有了这样可以对口型播放的视频，我们就可以再改造一下刚才通过 Gradio 创建的应用，不要光让机器人用语音了，直接用视频来开口说话吧。

我们在前面语音聊天界面的基础上，又做了几处改造。

1. 我们在原有的 Gradio 界面中，又增加了一个 HTML 组件，显示头像图片，并用来播放对好口型的视频。默认一开始，显示的是一张图片。


 复制代码

```
1 .....
2     with gr.Row():
3         video = gr.HTML(f'<source
7         history.append(response)
8         responses = [(u,b) for u,b in zip(history[::2], history[1::2])]
9         return responses, video_html, history
10    else:
11        video_html = f'<img src="{avatar_url}" width="320" height="240" alt="John
12        responses = [(u,b) for u,b in zip(history[::2], history[1::2])]
13        return responses, video_html, history
```

注：通过 ChatGPT 获取回答，然后将回答和头像一起生成一个视频文件自动播放。

3. 在获取视频的时候需要注意一点，就是我们需要等待视频在 D-ID 的服务器生成完毕，才能拿到对应的 result\_url。其实更合理的做法是注册一个 webhook，等待 d-id 通过 webhook 通知我们视频生成完毕了，再播放视频。不过考虑到演示的简便和代码数量，我

们就没有再启用一个 HTTP 服务来接收 webhook，而是采用 sleep 1 秒然后重试的方式，来实现获取视频的效果。

 复制代码

```
1 def get_mp4_video(input, avatar_url=avatar_url):
2     response = generate_talk(input=input, avatar_url=avatar_url)
3     talk = get_a_talk(response['id'])
4     video_url = ""
5     index = 0
6     while index < 30:
7         index += 1
8         if 'result_url' in talk:
9             video_url = talk['result_url']
10            return video_url
11        else:
12            time.sleep(1)
13            talk = get_a_talk(response['id'])
14    return video_url
```

注：result\_url 字段会在服务器端把整个视频生成完成之后才出现，所以我们需要循环等待。

改造完整体代码如下：

 复制代码

```
1 import openai, os, time, requests
2 import gradio as gr
3 from gradio import HTML
4 from langchain import OpenAI
5 from langchain.chains import ConversationChain
6 from langchain.memory import ConversationSummaryBufferMemory
7 from langchain.chat_models import ChatOpenAI
8
9 openai.api_key = os.environ["OPENAI_API_KEY"]
10
11 memory = ConversationSummaryBufferMemory(llm=ChatOpenAI(), max_token_limit=2048)
12 conversation = ConversationChain(
13     llm=OpenAI(max_tokens=2048, temperature=0.5),
14     memory=memory,
15 )
16
17 avatar_url = "https://cdn.discordapp.com/attachments/1065596492796153856/10956174
18
```



```

19 def generate_talk(input, avatar_url,
20                 voice_type = "microsoft",
21                 voice_id = "zh-CN-YunyeNeural",
22                 api_key = os.environ.get('DID_API_KEY')):
23     url = "https://api.d-id.com/talks"
24     payload = {
25         "script": {
26             "type": "text",
27             "provider": {
28                 "type": voice_type,
29                 "voice_id": voice_id
30             },
31             "ssml": "false",
32             "input": input
33         },
34         "config": {
35             "fluent": "false",
36             "pad_audio": "0.0"
37         },
38         "source_url": avatar_url
39     }
40     headers = {
41         "accept": "application/json",
42         "content-type": "application/json",
43         "authorization": "Basic " + api_key
44     }
45
46     response = requests.post(url, json=payload, headers=headers)
47     return response.json()
48
49
50
51 def get_a_talk(id, api_key = os.environ.get('DID_API_KEY')):
52     url = "https://api.d-id.com/talks/" + id
53     headers = {
54         "accept": "application/json",
55         "authorization": "Basic " + api_key
56     }
57     response = requests.get(url, headers=headers)
58     return response.json()
59
60
61
62 def get_mp4_video(input, avatar_url=avatar_url):
63     response = generate_talk(input=input, avatar_url=avatar_url)
64     talk = get_a_talk(response['id'])
65     video_url = ""
66     index = 0
67     while index < 30:

```

```

68         index += 1
69         if 'result_url' in talk:
70             video_url = talk['result_url']
71             return video_url
72         else:
73             time.sleep(1)
74             talk = get_a_talk(response['id'])
75     return video_url
76
77 def predict(input, history=[]):
78     if input is not None:
79         history.append(input)
80         response = conversation.predict(input=input)
81         video_url = get_mp4_video(input=response, avatar_url=avatar_url)
82         video_html = f"<video width='320' height='240' controls autoplay><source
83         history.append(response)
84         responses = [(u,b) for u,b in zip(history[::2], history[1::2])]
85         return responses, video_html, history
86     else:
87         video_html = f'

👍 10





**John**

2023-04-24 来自加拿大

这个paddleBoBo都一年没更新啦 还有没有平替或者潜在新产品呢

作者回复: PaddleBobo其实没有几行代码, 本质上就是 Deepfake类的GAN的解决方案, 开源的GAN的库都可以看看是否适合作为平替。

共 2 条评论 >



1



**John**

2023-04-24 来自加拿大

现在HeyGen不错 就是收费不低

作者回复: 嗯, 现在数字人类的产品都不便宜



1



**劉仲仲**

2023-04-28 来自广东

出现error:module 'pexpect' has no attribute 'spawn',已经是最新的pexpect

作者回复: PaddleGAN应该还不支持windows

共 2 条评论 >



**abc** 😊

2023-04-25 来自福建

老师, 如果想要AI学习我的写作风格, 按照我的风格写作, 要怎么训练呢?

作者回复: 用18讲fine-tune的方式, 输入大量你自己写作的语料

但是fine-tune对于数据量还是有一定要求的, 至少有个500篇的文章才有一定效果吧。

共 2 条评论 >



**粉墨之下**

2023-04-24 来自甘肃

本地运行后，回复时报错：Retrying langchain.llms.openai.completion\_with\_retry.<locals>.\_completion\_with\_retry in 4.0 seconds as it raised APIConnectionError: Error communicating with OpenAI: HTTPSPool(host='api.openai.com', port=443): Max retries exceeded with url: /v1/completions (Caused by NewConnectionError('<urllib3.connection.HTTPSConnection object at 0x000001F9C7DAD670>: Failed to establish a new connection: [WinError 10060] 由于连接方在一段时间后没有正确答复或连接的主机没有反应，连接尝试失败。'))).

作者回复：大概率是神奇的网络访问问题，这个需要自己想办法解决啦，或者直接通过Colab环境来运行。



**Geek\_4ec46c**

2023-04-24 来自福建

刚看了下,这个did的价格不是一般的贵....

作者回复：对，国内的数字人现在报价也比较贵，有数据的话，自己通过GAN来做梗合适一些。

共 2 条评论 >

