如何使用TensorFlow实现音频分类任务

By 机器之心2017年12月16日 11:06

> 本文介绍了一种使用 TensorFlow 将音频进行分类（包括种类、场景等）的实现方案，包括备选模型、备选数据集、数据集准备、模型训练、结果提取等都有详细的引导，特别是作者还介绍了如何实现 web 接口并集成 IoT。

简介

有很多不同的项目和服务能够识别人类的语音，例如 Pocketsphinx、Google』s Speech API，等等。这些应用和服务能够以相当好的性能将人类的语音识别成文本，但是其中却没有一个能够分得清麦克风捕捉到的是哪一种声音：人声、动物声音或者音乐演奏声。

我们面临这个任务的时候，就决定去调研一下，并开发一个能够使用机器学习算法来区分声音的示例项目。这篇文章具体描述了我们选择哪款工具、我们面临的挑战是什么、我们如何用 TensorFlow 训练模型，以及如何运行我们的开源项目。为了把它们用在给第三方应用提供的云服务上，我们还在 DeviceHive 和 IoT 平台上提供了识别结果。

选择工具和分类模型

首先我们需要选择一些能够运行神经网络的软件。我们发现的第一个合适的解决方案是 Python Audio Analysis。

机器学习中的主要问题是要有一个好的训练数据集。对于音乐分类和语音识别而言，有很多数据集，但是并没有多少数据集是用来做随机声音分类的。经过调查，我们发现了 urban sound dataset（https://serv.cusp.nyu.edu/projects/urbansounddataset/）这个数据集。

经过一些测试之后，我们面临着以下问题：

- pyAudioAnalysis 不够灵活。它的参数种类参数太少，并且一些参数的计算是不受控制的，例如，训练实验的数量是基于样本数量的，你不能通过 pyAudioAnalysis 改变它。
- 这个数据集只有 10 个种类，而且它们都是「urban」类型。

我们发现的另一个解决方案是 Google AudioSet，它是基于有标签的 YouTube 视频片段，可以以两种格式下载：

- 每一个视频片段都有 CSV 文件描述，包括 YouTube 视频 ID、起始时间和结束时间、以及一个或多个标签。
- 提取出的音频特征以 TensorFlow Record 文件的形式被存储。

这些特征和 YouTube-8M 模型是兼容的。这个解决方案也提供了 TensorFlow VGGish 模型作为特征提取器。它满足了我们的大部分需求，因此也就成为了我们的最佳选择。

训练模型

下一个任务就是了解 YouTube-8M 接口是如何运行的。它是被设计来处理视频的，但是幸运的是它也能够处理音频。这个库是相当方便的，但是它有固定的样本类别数。所以我们对它进行了小小的修改，以将类别数作为参数传入。

YouTube-8M 能够处理两种类型的数据：总体特征和帧特征。Google AudioSet 能够将我们之前提到的数据作为特征。通过研究之后我们发现那些特征是以帧的格式给出的。接下来，我们需要选择要被训练的模型。

资源、时间和精度

GPU 比 CPU 更适合做机器学习。你可以在这里看到更多的相关信息（https://docs.devicehive.com/blog/using-gpus-for-training-tensorflow-models）。所以我们跳过这个过程直接开始实验设置。我们在实验中使用的是一台装有 4GB 显存的 NVIDIA GTX 970 的 PC。

在我们的案例中，训练时间并不十分重要。只需要 1 到 2 小时就足以做出关于模型选择和准确率的初步决定。

当然，我们想尽可能得到更好的准确率。但是，为了训练出一个更复杂的模型（有潜力得到更高的准确率），你需要更大的 RAM（当然对于 GPU 而言就是显存了）。

选择模型

这里是具有细节描述的 YouTube-8M 模型完整列表（https://github.com/google/youtube-8m#overview-of-models）。因为我们的训练数据是帧格式的，所以必须使用帧级别的模型。Google AudioSet 数据集为我们提供的数据被分成了三部分：均衡的训练集、不均衡的训练集以及评估集。

还有一个修正版的用于训练和评估的 YouTube-8M 模型。可以在这里获得：（https://github.com/igor-panteleev/youtube-8m）

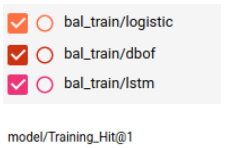
均衡的训练集

训练命令如下：

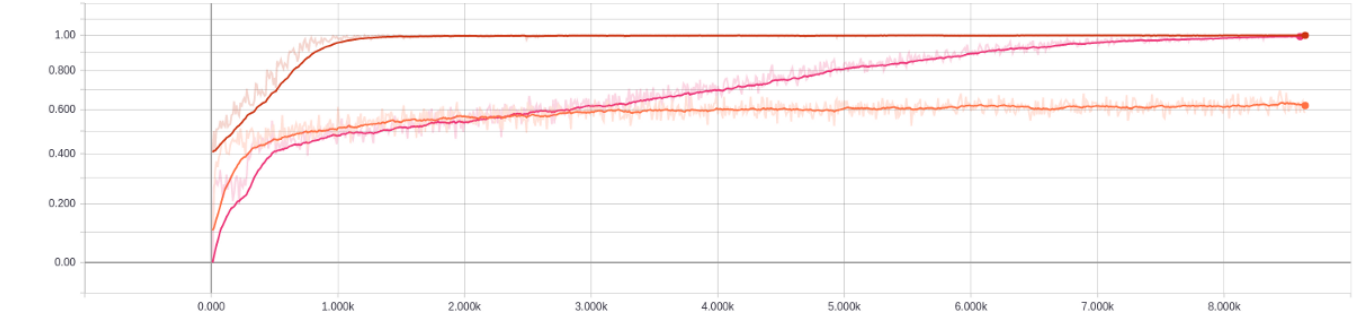
1. python train.py –train_data_pattern=/path_to_data/audioset_v1_embeddings/bal_train/*.tfrecord –num_epochs=100 –learning_rate_decay_examples=400000 –feature_names=audio_embedding –feature_sizes=128 –frame_features –batch_size=512 –num_classes=527 –train_dir=/path_to_logs –model=ModelName
2.

按照文档的建议，我们将 LSTM 模型的基础学习率改为 0.001。此外，我们还将 LSTM 单元的默认大小改为 256，因为我们没有足够大的 RAM。

现在我们看一下训练结果：



model/Training_Hit@1

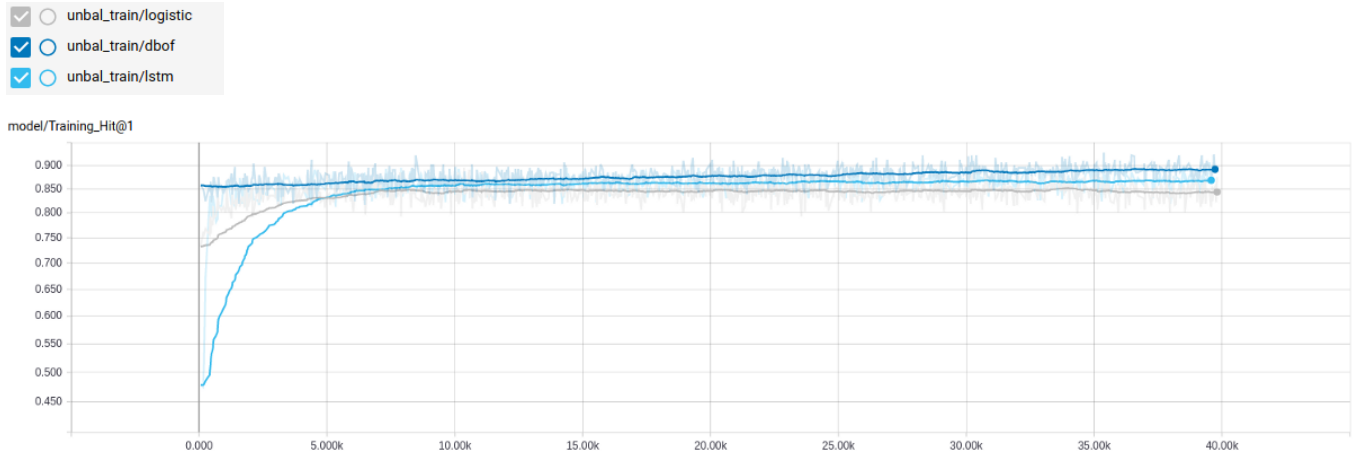


模型名	训练时间	最后一次迭代的准确率	平均准确率
Logistic	14m 3s	0.58590	0.5560
Dbof	31m 46s	1.000	0.5220
LSTM	45m 53s	0.9883	0.4581

如上所示，我们在训练阶段得到了较好的结果，但是并不意味着在测试的时候也能得到同样好的结果。

不均衡训练

让我们来试试不均衡的数据集吧。它有更多的样本，所以我们将 epochs 的数目改为 10（最小应该改成 5，因为训练会耗费很多的时间）。



模型名	训练时间	最后一次迭代的准确率	平均准确率
Logistic	2h 4m 14s	0.875500.5560	0.5125
Dbof	4h 39m 29s	0.8848	0.5605

LSTM	9h 42m 52s	0.8691	0.5396

训练日志

如果你想核查我们的训练日志，可以在这里下载 (https://s3.amazonaws.com/audioanalysis/train_logs.tar.gz)，然后运行：

1. tensorboard –logdir /path_to_train_logs/
2.

然后在浏览器中打开主机 6006 端口就可以看到了。

关于训练的更多内容

YouTube-8M 采用了很多参数，大部分都会影响训练过程。例如：你可以调节学习率或者 epochs 的数量，这两个参数能够很明显的改变训练过程。还有 3 个用来计算损失的函数，以及很多其他有用的变量，你可以改变它们来提升结果。

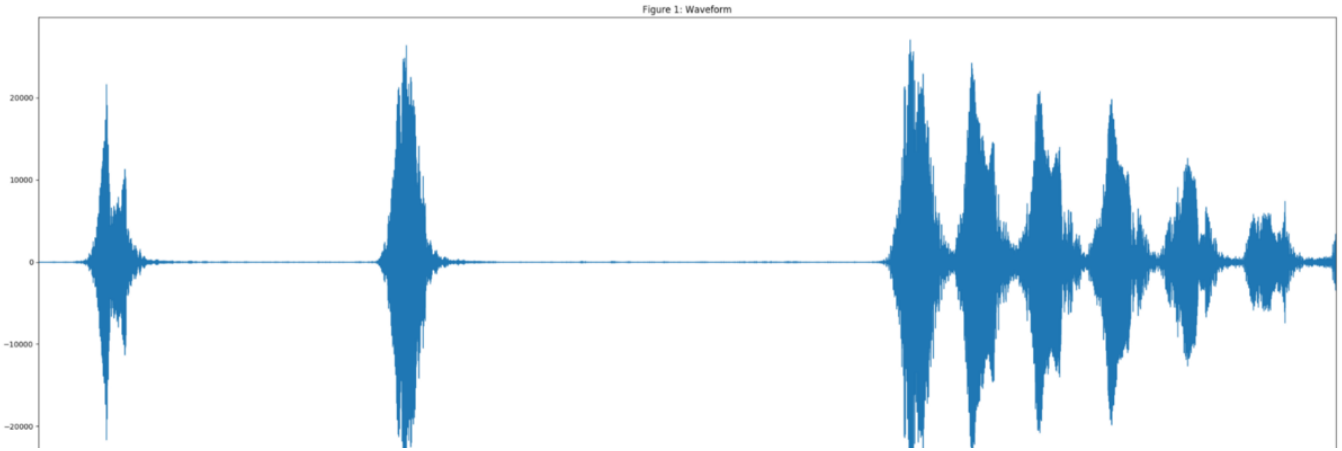
现在我们已经有了一些训练好的模型，是时候添加一些代码与它们交互了。我们需要从麦克风采集音频。这里我们使用 PyAudio，它提供了可以在很多平台上运行的简单接口。

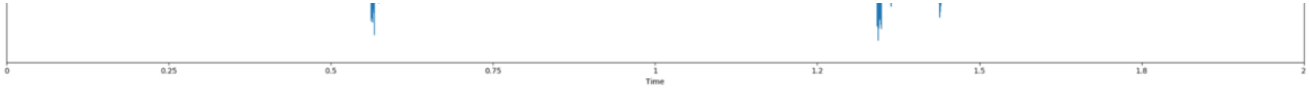
音频准备

正如我们之前所提及的，我们要使用 TensorFlow 的 VGGish 模型作为特征提取器。这里是转换过程的一个简短解释：

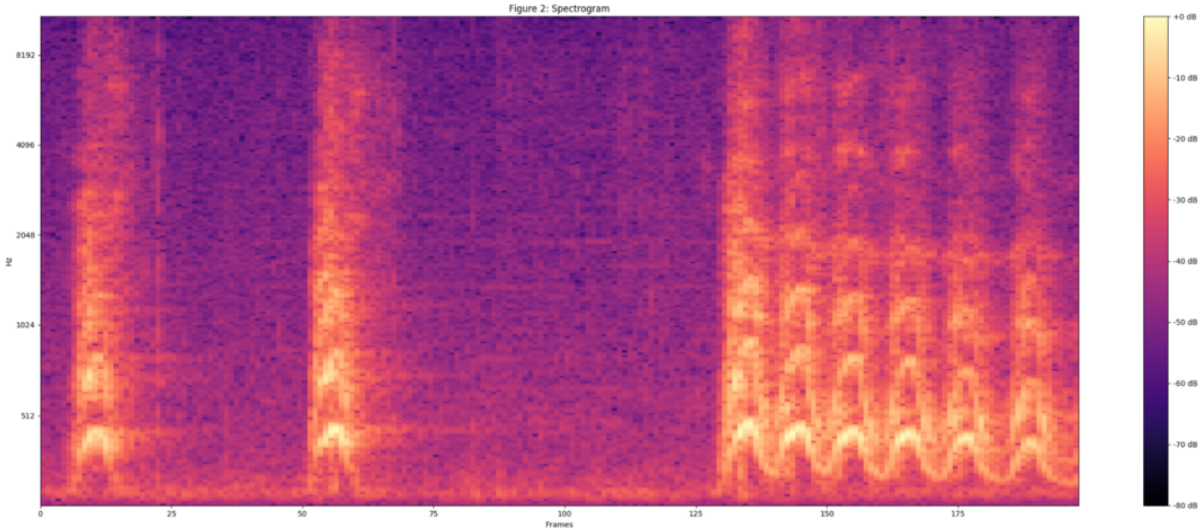
用 UrbanSound 数据集中「犬吠」声音样例来作为可视化的例子。

将音频重采样为 16kHz 单声道。

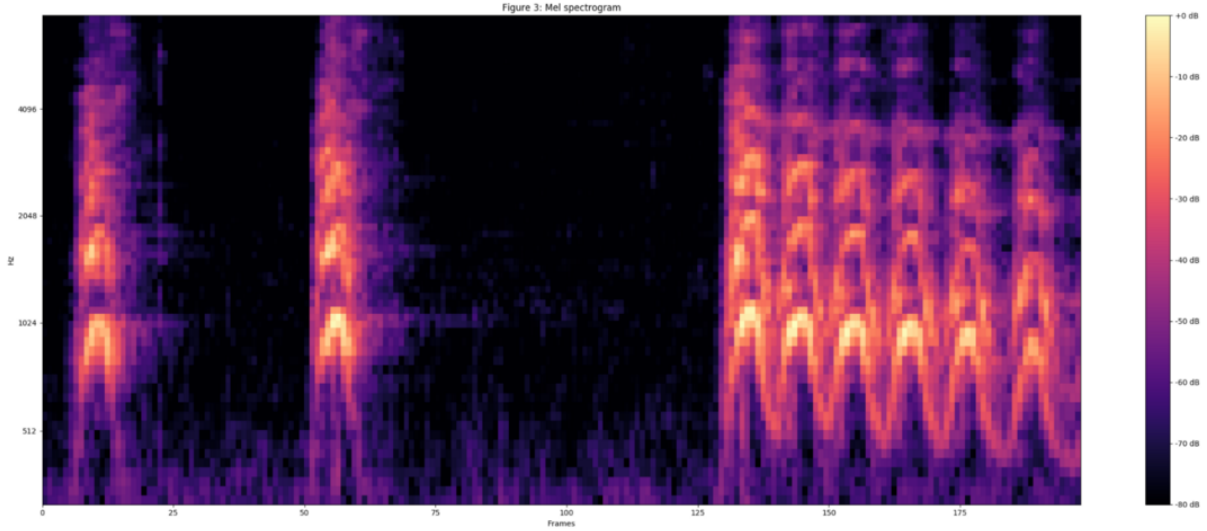




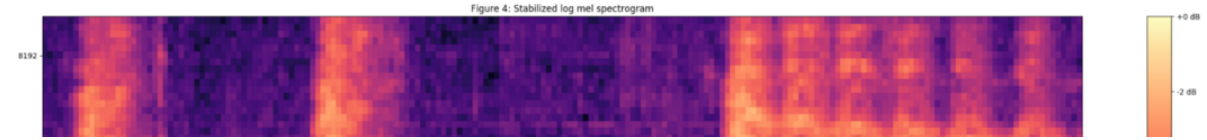
使用 25ms 的帧长、10ms 的帧移，以及周期性的 Hann 窗口对语音进行分帧，对每一帧做短时傅里叶变换，然后利用信号幅值计算声谱图。

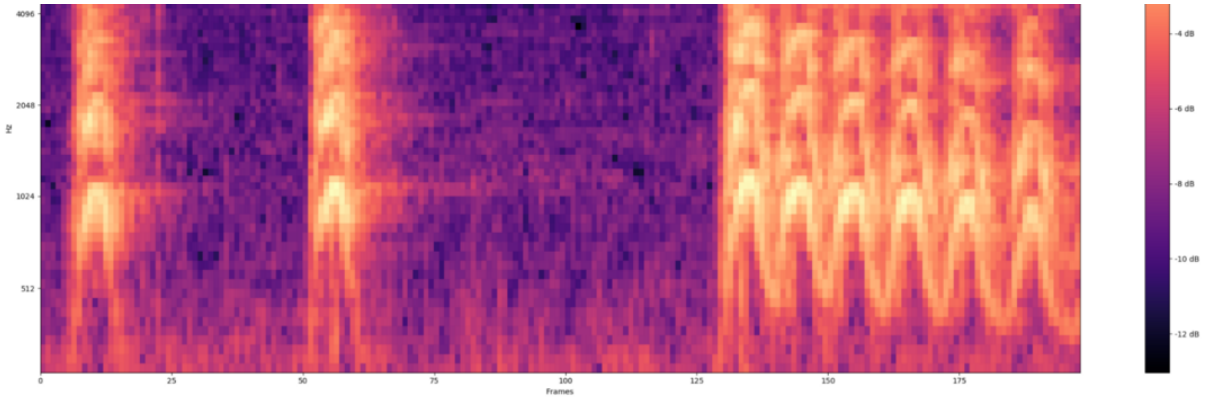


通过将声谱映射到 64 阶 mel 滤波器组中计算 mel 声谱。



计算 log(mel-spectrum + 0.01)，得到稳定的 mel 声谱，所加的 0.01 的偏置是为了避免对 0 取对数。





然后这些特征被以 0.96s（这段「犬吠」声的总时长）的时长被组帧，并且没有帧的重叠，每一帧都包含 64 个 mel 频带，时长 10ms（即总共 96 帧）。

然后这些样本被输入到 VGGish 模型中以提取特征向量。

分类

最后我们需要一个能够把数据输入到神经网络的接口，以得到分类结果。

我们使用 YouTube-8M 作为一个例子，但是会做一些修改，去掉序列化/反序列化（serialization/deserialization）步骤。

可以在这里看到我们的结果（https://github.com/devicehive/devicehive-audio-analysis）。

安装

PyAudio 使用 libportaudio2 和 portaudio19-dev，所以在安装 PyAudio 之前需要先安装这两个工具。

还需要一些 python 库，你可以使用 pip 来安装它们。

1. pip install -r requirements.txt
2.

你还需要下载并提取具有保存好的模型的项目主目录，可以在这里找到（https://s3.amazonaws.com/audioanalysis/models.tar.gz）。

运行

我们的项目提供了 3 个可供使用的接口。

1. 处理录制好的音频文件

只需要运行下面的命令就可以：

```
1. python parse_file.py path_to_your_file.wav
2.
```

然后你就会在终端中看到以下信息：语音：0.75；音乐：0.12；在大厅里面：0.03。

这个结果由输入的文件决定。这些值是神经网络做出的预测。数值越大，则说明输入文件中的音频属于该类别的概率比较大。

2. 从麦克风中捕捉并处理数据

运行 python capture.py 开始从麦克风中无限制地采集数据。默认配置下，它会每 5-7s 将数据输入到神经网络。可以在其中看到之前例子的结果。

在这个案例中，你可以使用–save_path=/path_to_samples_dir/运行上面的命令，然后所有采集到的数据都会以 wav 文件的格式存储在你提供的路径中。当你想用同样的样本尝试不同的模型时，这个函数是很有用的。可以使用-help 来获取更多的信息。

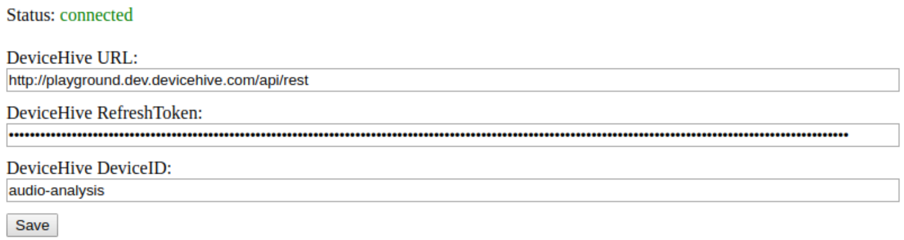
3.web 接口

python daemon.py 实现了一个简单的 web 接口，默认配置下在本地的 8000 端口（http://127.0.0.1:8000/）。然后我们使用之前例子中提到的同样的代码。你可以看到对声音时间的最后 10 次预测（http://127.0.0.1:8000/events）：

```
2017-11-02 17:29:38  Music: 0.26
2017-11-02 17:29:43  Music: 0.56, Wind chime: 0.20, Speech: 0.13, Chime: 0.12
2017-11-02 17:29:48  Music: 0.51, Wind chime: 0.10
2017-11-02 17:29:53  Music: 0.68, Spray: 0.15, Grunge: 0.12, Speech: 0.11
2017-11-02 17:29:58  Music: 0.56, Speech: 0.13
2017-11-02 17:30:03  Music: 0.56, Speech: 0.15, Snort: 0.13, Crunch: 0.10
2017-11-02 17:30:08  Speech: 0.28, Music: 0.23, Fly, housefly: 0.23, Bee, wasp, etc.: 0.14
2017-11-02 17:30:13  Speech: 0.33, Music: 0.24, Fly, housefly: 0.18, Bee, wasp, etc.: 0.11
2017-11-02 17:30:18  Fly, housefly: 0.28, Speech: 0.26, Music: 0.19, Bee, wasp, etc.: 0.17
2017-11-02 17:30:23  Speech: 0.21, Music: 0.15
```

IoT 服务集成

最后但是也是比较重要的一个：集成在 IoT 基础设施中。如果你运行了我们前面提到的 web 接口，你可以在索引页面上看到 DeviceHive 客户状态和配置。只要客户端已经连接成功了，预测结果会以通知的形式发送到特定的设备上。



结论

正如你所看到的，TensorFlow 是一款非常灵活的工具，在很多机器学习应用中都很有用，例如图像处理和语音识别。将这个方法和物联网平台结合起来可以让你在很多领域建立智能解决方案。

智慧城市可以将这个解决方案用于安全目的，可以持续地监听玻璃破碎声、枪声以及其他与犯罪相关的声音。甚至在雨林中，可以用这个方法通过分析它们的声音来跟踪动物和鸟类。

物联网设备可以收到所有这类通知。这个解决方案可以安装在本地设备上以最小化通信和云成本（尽管也可以部署在云端作为一种云服务），并且可被定制成只接收除包含原始音频之外的信息。别忘了，这是一个开源的项目，随意使用吧。

原文地址：https://medium.com/iotforall/sound-classification-with-tensorflow-8209bdb03dfb

TensorFlow工程NLP分类

联系电话：+86 010-57150141

联系邮箱：contact@jiqizhixin.com