Theory of combined 2 neural network

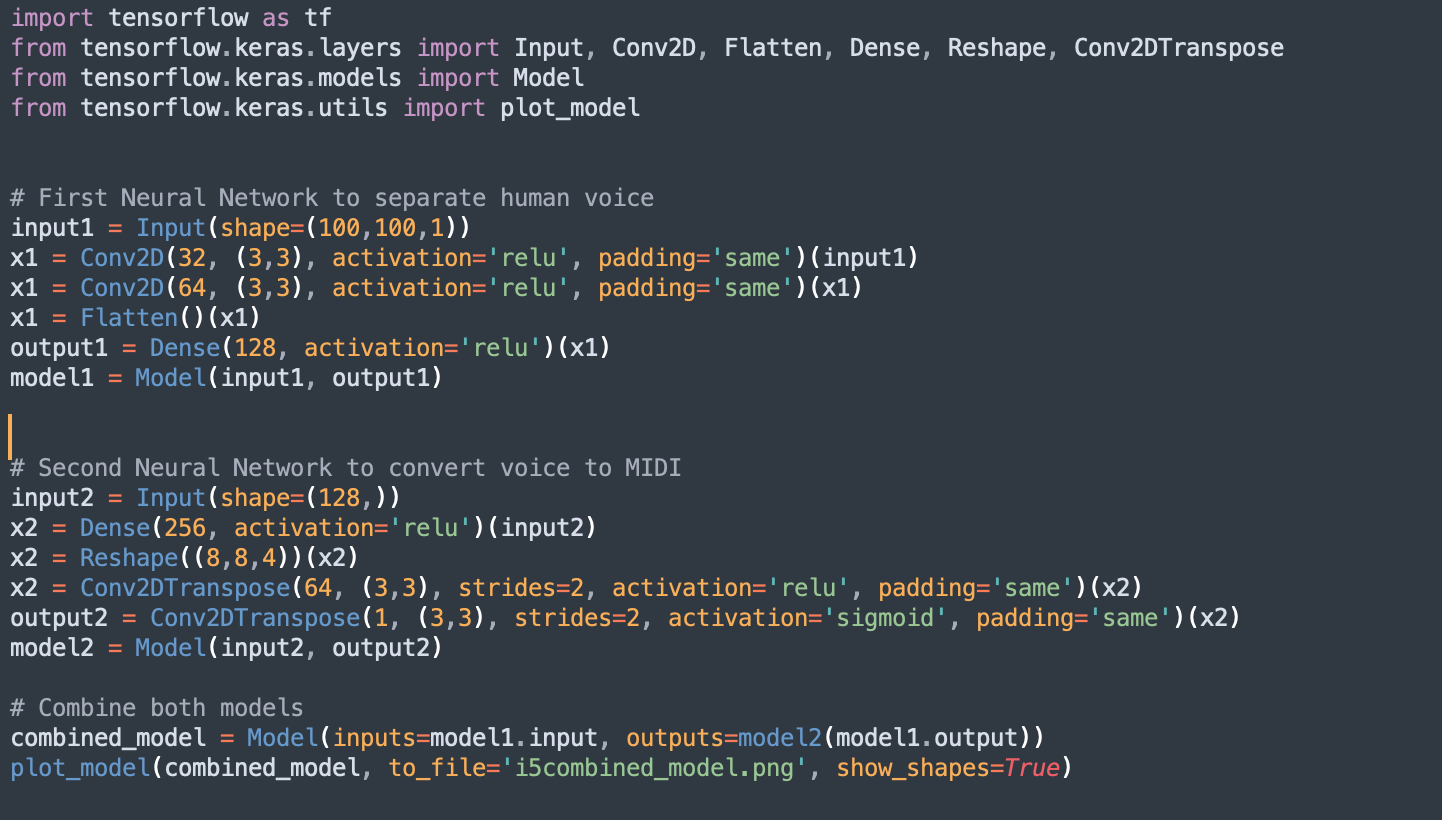
abel - 2023年2月18日

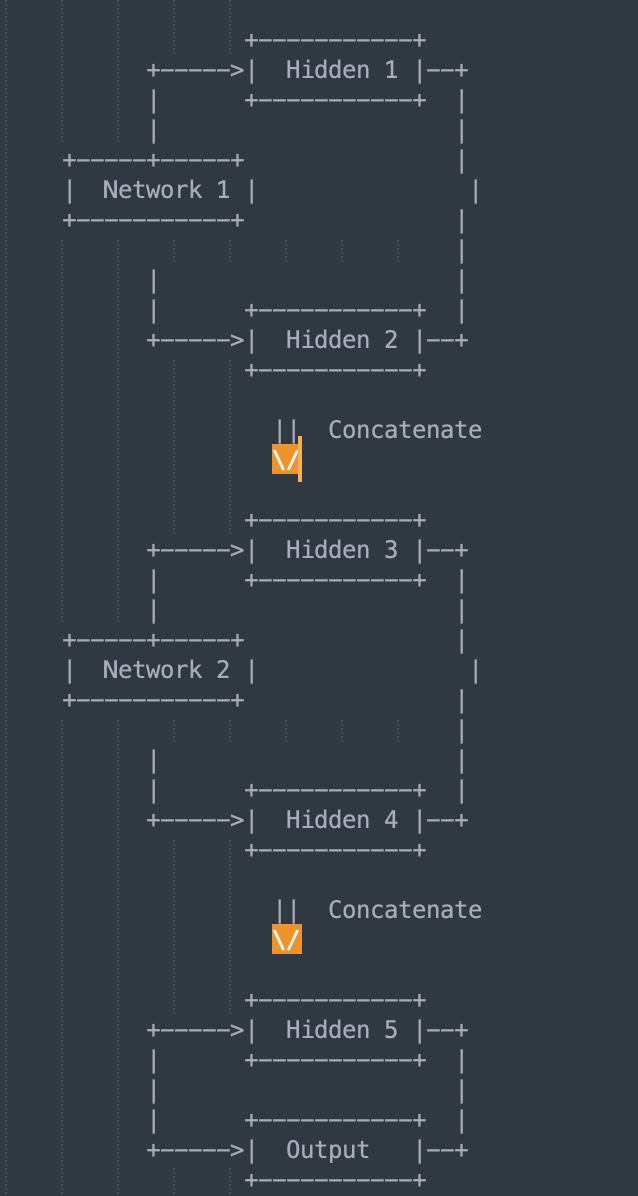


介绍

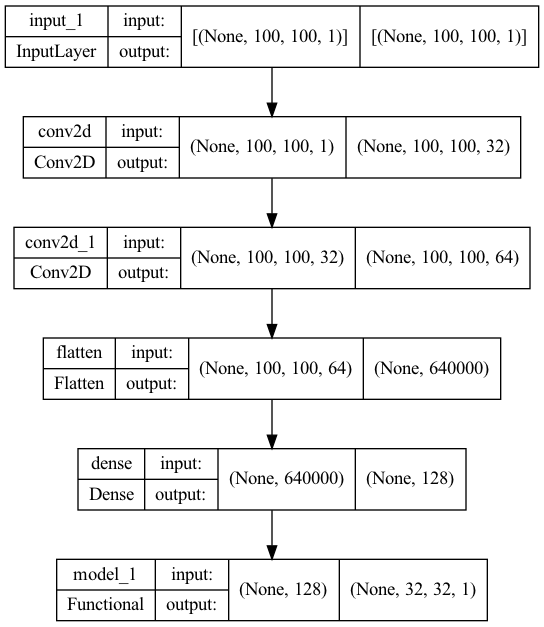
Nework1 should be speeter network

Nework2 should be Volcan network(or other like network)





I write some prototype code of combined the 2 networks:



具体来说，我们可以设计一个两层神经网络结构，第一层网络是一个spleeter神经网络，用于分离声音中的人声，

第二层网络是一个vocano网络或者卷积神经网络，用于将分离出的人声转换成MIDI格式。

以下是具体的方案：

第一层网络：卷积神经网络 对于我们的项目就是spleeter的Wave-U-Net

是一种基于卷积神经网络（CNN）和递归神经网络（RNN）的深度学习模型

输入：声音信号（例如，一个.wav文件）

输出：人声和背景噪声

训练数据：包括人声和背景噪声的音频文件

第二层网络：循环神经网络或卷积神经网络，对于我们是vocano的卷积神经网络（CNN）和递归神经网络（RNN）

Melody Line模型主要由卷积神经网络和循环神经网络组成，用于提取唱歌旋律的特征表示和时序建模；

Non-Melody模型则是由一系列卷积神经网络和全连接层组成，用于检测非唱歌声音。

两个子模型的输出结果可以结合起来，形成最终的旋律线估计结果。

输入：人声信号

输出：MIDI格式的音符序列

训练数据：包括人声信号和对应的MIDI文件

在这种架构中，第一层网络专门用于处理声音信号，将声音分离成人声和背景噪声，

这样第二层网络就可以专注于处理人声信号，将它转换成MIDI格式的音符序列。

这种分层设计可以帮助我们更好地处理复杂的任务，并且可以在不同的层级上分别训练神经网络，从而获得更好的性能和效果

代码解释

**Spleeter** 代码解释

dataset.py：这个文件定义1个类主要是用来管理音频文件的数据集的，它可以读取、加载和预处理音频数据集，并将其分成训练集和测试集。

\_\_main\_\_.py：这个文件是整个程序的入口，它主要是用来解析命令行参数，并根据不同的参数运行不同的功能。例如，你可以通过这个文件来运行音频文件的分离、训练神经网络、评估模型的性能等。

separator.py：这个文件定义了一个 Separator 类，这个类的主要功能是将一段音频文件分离成多个音轨（如人声、吉他、鼓等）。它使用了深度学习算法来进行音频信号的分离，具体的实现细节可以不用关心。

types.py：这个文件定义了一些自定义类型，用于表示音频数据和神经网络模型。这些类型主要是为了提高代码的可读性和可维护性，使代码更加清晰易懂。

这几个文件主要用来实现音频文件分离的，

dataset.py 主要是用来管理数据集， \_\_main\_\_.py 用来解析命令行参数并运行不同的功能，

separator.py 实现了音频文件分离的核心算法，

types.py 则定义了一些自定义类型

**Utils**下面的文件：

configuration.py: 这个文件中定义了一个配置类，它用于读取和解析一个配置文件。

配置文件中包含了一些参数和选项，这些参数和选项可以用来控制算法的行为。这个配置类提供了一些方法，用于读取和设置这些参数和选项。

**pleeter/model**：

这块用于音频分离的 Python 代码，使用 TensorFlow 库进行数学计算。它包含了一些类和函数，用于处理音频数据并将其输入到预定义的模型中，以获得多轨道输出，这些类和函数的名称可能有些复杂，但是它们的作用非常简单。它们主要有三个部分：

用于导入必要的库和函数。其中，tf.compat.v1.placeholder 是一个 TensorFlow 函数，用于创建一个占位符，它允许我们在稍后的计算中提供实际数据。

get\_model\_function 函数的作用是根据传入的模型类型获取 TensorFlow 函数，以便应用于输入张量。这个函数使用 Python 的 importlib 模块动态地导入指定的 Python 模块，并从中获取指定的函数。

InputProvider 类和其两个子类 WaveformInputProvider 和 SpectralInputProvider 的作用是为模型提供输入数据。它们的主要区别在于处理输入数据的方式。WaveformInputProvider 用于接收原始音频波形数据，SpectralInputProvider 则用于接收音频的频谱表示。这两个类中的方法用于创建 TensorFlow 占位符，并为模型提供适当的数据。InputProviderFactory 类则用于创建适当的输入提供程序对象，具体取决于所选择的频谱表示方法。

总结就是：这些代码主要用于处理音频数据，并使用 TensorFlow 库进行数学计算，以从输入音频中分离出多个独立的音轨

################################################

**vocano/**代码解释

**core.py**

主要是提取歌声的特征和旋律，然后使用两个预训练模型来预测歌曲的旋律。最后，它会生成MIDI文件和合成音频文件。以下是每个函数的作用：

\_\_init\_\_(self, args)：初始化实例，并选择要使用的设备（CPU或GPU）。

\_select\_device(self)：选择要使用的设备。

\_free\_gpu(self)：释放GPU使用的内存。

\_download\_from\_googledrive(self, file\_id, dest\_path)：从Google Drive下载模型文件。

download\_ckpt(self)：下载需要的预训练模型文件。

transcription(self)：从输入文件中提取特征和旋律，预测歌曲的旋律，生成MIDI文件和合成音频文件，并将它们保存到磁盘上。

data\_preprocessing(self)：提取音频文件中的特征和旋律。

feature\_extraction(self, wave\_dir, use\_cupy)：提取音频文件的特征和旋律。

load\_model(self)：加载预训练模型文件。

voice\_transcription(self)：使用预测模型预测歌曲的旋律。

gen\_midi(self)：根据预测的旋律生成MIDI文件。

gen\_wav(self)：合成预测的旋律和音频文件。

save\_midi(self, midi\_data, midi\_dir)：将MIDI数据保存到磁盘上。

save\_wav(self, wave\_data, wave\_dir)：将合成的音频数据保存到磁盘上

**preprocess.py**

就是主要传递各种参数进入给调用主程序

**transcription.py**

Python内置的 argparse 模块来解析命令行参数，并将这些参数传递给 SingingVoiceTranscription 类的实例 solver，

然后调用实例的 transcription() 方法来运行整个程序的主要逻辑。这个程序的主要功能是从一个给定的wav文件中提取声乐旋律，

并生成相应的midi文件和wav文件。其中的参数可以用来控制一些设置，例如生成文件的名称、存储目录、设备类型、使用GPU加速等

**model/Patch\_CNN.py**

KitModel类：该类定义了一个名为KitModel的神经网络模型，并继承了nn.Module类，它包括一个初始化方法和一个前向传递方法。

初始化方法：该方法加载了权重字典并定义了神经网络的各个层，包括卷积层、全连接层等。

前向传递方法：该方法实现了神经网络的前向传递过程，包括卷积、ReLU激活函数、dropout、池化、全连接、softmax等操作。

这块主要用于训练和测试图像分类模型，其中卷积层、池化层、全连接层等是深度学习中常用的操作，ReLU激活函数、dropout、softmax等是常见的神经网络技巧。

**model/PyramidNet\_ShakeDrop.py**

这段代码定义了一个名为PyramidBlock的神经网络模块。这个模块接受一个输入张量x，然后通过一些卷积和批量标准化等操作来计算输出张量h。具体来说，模块分为两个分支：主分支和shortcut分支。主分支由两个卷积层和两个批量标准化层构成。

主分支的目的是提取输入张量的特征，并将其转换为输出张量h。在这个过程中，还有一个可选的ShakeDrop层，可以帮助增强模块的泛化能力

################################################

自己写的**i2polyphonic\_test.py**

Librosa是一个用于音频、音乐分析、处理的python工具包，一些常见的时频处理、特征提取、绘制声音图形等功能应有尽有，功能十分强大。LibROSA 是一个用于音乐和音频分析的 python 包。它提供了创建音乐信息检索系统所需的构建块。

i2polyphonic\_test.py代码的第1部分中，使用了 Wave 和 Librosa 两个库来读取 WAV 文件并创建新的 WAV 文件。num\_channels 表示音频的通道数，sample\_width 表示每个采样点占据的字节数，frame\_rate 表示采样率，num\_frames 表示总采样点数。在创建新的 WAV 文件时，代码通过复制原始音频数据的方式实现多音轨效果。

i2polyphonic\_test.py在第2部分中（偶默认注释），使用了 Librosa 库来处理 WAV 文件，并将其转换为 MIDI 文件。首先加载 WAV 文件，并使用 chroma\_stft 函数来提取音频文件的色度特征，即将音频信号转换为一组音符。接下来对色度特征进行规范化处理，以便在 MIDI 文件中正确地表示音符的时值和音高。然后创建一个新的 MIDI 并添加时间签名、节奏和音符事件。

代码根据色度特征来生成音符序列，并将它们添加到 MIDI 文件中。最后，将新的 MIDI 文件保存在本地系统