Ses Parçalarını Ayırmak İçin Python ve Sinir Ağlarını Kullanma

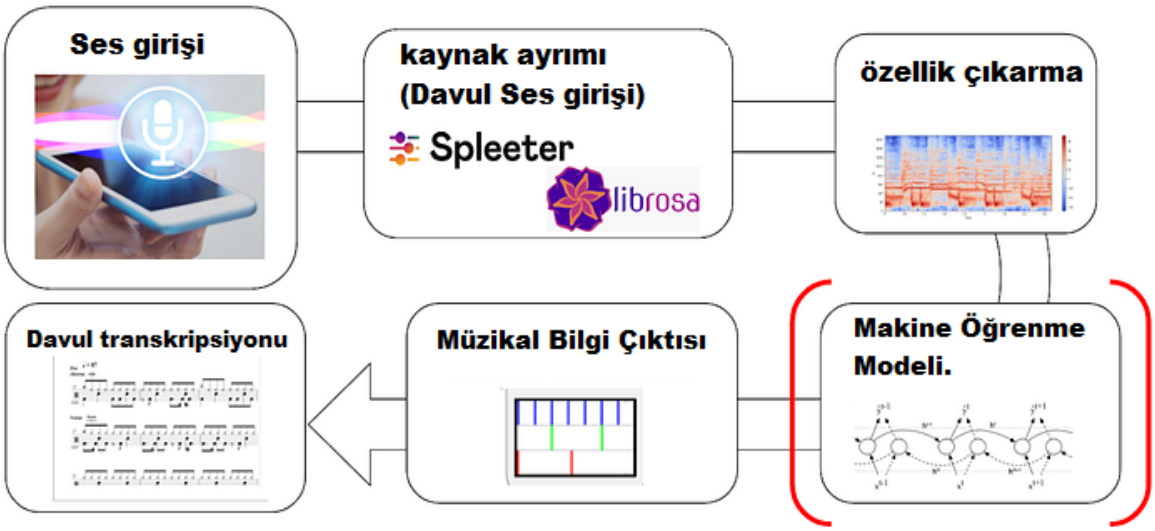
Merak ettiniz mi hiç, prodüktörler ve DJ'ler nasıl olağanüstü remixler oluşturuyorlar, yani bir şarkıdan belirli enstrümanları ayrıştırarak onları yeni bir parçaya dönüştürüyorlar? İşte AI günümüzünde bunu başarmak için, Python'da Convolutional Neural Networks (CNNs) ve **Recurrent Neural Networks** (RNNs) kullanılıyor!

Müzik hayatımızın büyük bir parçasıdır ve stereo müzik hepinizin sevdiği popüler bir biçimdir. Fakat bir parçadan belirli enstrümanları çıkarmak istediniz mi hiç?

O zaman, CNN ve RNN nedir? CNN'ler, görüntü sınıflandırması için kullanılan bir tür sinir ağıdır, ancak ses işleme için de kullanılabilir. RNN'ler, konuşma tanıma gibi veri dizilerini tahmin etmek için kullanılan başka bir tür sinir ağıdır. Bu iki tür sinir ağını birleştirerek, kaynak ayırma olarak adlandırılan bir teknik kullanarak, bir stereo parçadan belirli enstrümanları ayıklayabiliriz.

Başlamak için, **Librosa, NumPy, TensorFlow** veya **PyTorch** ve **Keras** gibi kütüphaneleri yüklemeniz gerekiyor. Bu kütüphaneler, ses dosyalarını yüklemenize, ses sinyallerini spektrogramlara dönüştürmenize, CNN ve RNN modelleri oluşturmanıza ve onları eğitmenize olanak sağlıyor.

Kütüphanelerinizi ayarladıktan sonra, Librosa kullanarak stereo ses dosyasını ve ayrı ayrı enstrüman parçalarını yükleyebilirsiniz. Bir sonraki adım, ses sinyalini frekans alanında ses sinyalinin 2D temsili olan bir büyüklük spektrogramına dönüştürmektir. Bunu yapmak için Librosa'nın "stft" işlevini kullanabilirsiniz.



Python, CNNs ve RNNs kullanarak stereo müzikten enstrümanları ayrıştırmak için kaynak ayrıştırmasını uygulamak için size yardımcı olmak için çeşitli kütüphaneler ve kaynaklar sunar.

Gerekli kütüphaneleri kurmak için, Python'da pip paket yöneticisi kullanabilirsiniz. Burada gerekli kütüphaneleri kurmak için bir örnek komut var.

```
PYpip install librosa numpy tensorflow kera
```

Ses dosyalarını yükle:

Veri kümesinden stereo ses dosyasını ve ona ait tekli enstrüman izlerini yüklemek için **librosa.load** fonksiyonunu kullanabilirsiniz. Burada bir örnek kod parçası var:

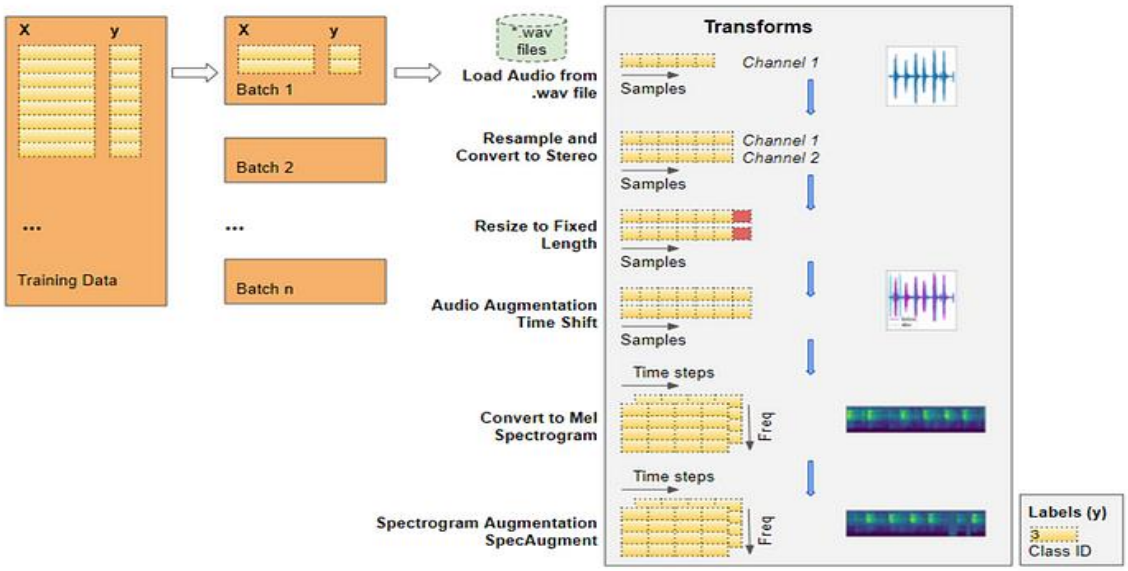
```
import librosa

# load stereo audio file
mix, sr = librosa.load('path/to/stereo/file.wav', sr=None, mono=False)

# load individual instrument tracks
vocals, sr = librosa.load('path/to/vocals/file.wav', sr=None, mono=True)
drums, sr = librosa.load('path/to/drums/file.wav', sr=None, mono=True)
bass, sr = librosa.load('path/to/bass/file.wav', sr=None, mono=True)
other, sr = librosa.load('path/to/other/file.wav', sr=None, mono=True)
```

Ses sinyalini ön işleme:

Karışık ses ve bireysel enstrüman izlerinden oluşan bir veri kümesi oluşturduktan sonra, modellerimizi eğitmek için kullanacağız. Ardından, Karışık ses sinyalinden özellikleri çıkarmak için bir CNN kullanabilir ve her bireysel enstrüman izinin büyüklük spektrogramını tahmin edebiliriz. Büyüklük spektrogramı, ses sinyalini frekans alanında temsil etmek için bir yoldur. Her enstrüman izinin büyüklük spektrogramını tahmin ederek, onları karışık ses sinyalinden ayırabiliriz.



Ses sinyalini büyüklük spektrogramına dönüştürmek için, librosa.stft fonksiyonunu kullanabilirsiniz. Burada bir örnek kod parçacığı var.

```
import numpy as np
import librosa
```

```
# convert audio signal to magnitude spectrogram
mix_spec = np.abs(librosa.stft(mix))# convert individual instrument tracks to magnitude
spectrograms
vocals_spec = np.abs(librosa.stft(vocals))
drums_spec = np.abs(librosa.stft(drums))
bass_spec = np.abs(librosa.stft(bass))
other_spec = np.abs(librosa.stft(other))
```

Veriyi hazırlayın:

Model için girdi ve çıktı verisi oluşturmak için, tek tek enstrüman izlerinin büyüklük spektrogramlarını 3D tensörün içine yığabilir ve karışık ses sinyalindeki büyüklük spektrogramını girdi olarak kullanabilirsiniz. Burada bir örnek kod parçacığı var:

```
# stack individual instrument tracks into a 3D tensor
y = np.stack([vocals_spec, drums_spec, bass_spec, other_spec], axis=-1)

# use magnitude spectrogram of mixed audio as input
X = mix_spec[..., np.newaxis]
```

CNN ve RNN modellerini oluşturun:

Keras API'sini kullanarak CNN ve RNN modellerini oluşturabilirsiniz. Burada bir örnek kod parçacığı var:

```
from keras.models import Model
from keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, LSTM,
Bidirectional

# define input shape
input_shape = X.shape[1:]

# create CNN model
inputs = Input(shape=input_shape)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
cnn_outputs = Dense(4, activation='linear')(x)
cnn_model = Model(inputs=inputs, outputs=cnn_outputs)

# create RNN model
inputs = Input(shape=(None, cnn_outputs.shape[1]))
x = Bidirectional(LSTM
(units=128, activation='relu', return_sequences=True))(inputs)
x = Bidirectional(LSTM(units=128, activation='relu', return_sequences=True))(x)
rnn_outputs = Dense(4, activation='linear')(x)
rnn_model = Model(inputs=inputs, outputs=rnn_outputs)
```

Modelleri eğitmek:

Keras API'sinin compile ve fit yöntemlerini kullanarak CNN ve RNN modellerini eğitebilirsiniz. Burada bir örnek kod parçacığı var:

```
# compile CNN model
cnn_model.compile(optimizer='adam', loss='mean_squared_error')

# fit CNN model
cnn_model.fit(X, y, batch_size=32, epochs=50, validation_split=0.2)

# compile RNN model
rnn_model.compile(optimizer='adam', loss='categorical_crossentropy')

# fit RNN model
rnn_model.fit(rnn_inputs, rnn_outputs, batch_size=32, epochs=50, validation_split=0.2)
```

Modelleri değerlendir:

Keras API'sinin değerlendirme yöntemini kullanarak CNN ve RNN modellerinin test kümesindeki performansını değerlendirebilirsiniz. Burada bir örnek kod parçacığı var:

```
# evaluate CNN model
cnn_loss = cnn_model.evaluate(X_test, y_test)

# evaluate RNN model
rnn_loss = rnn_model.evaluate(rnn_inputs_test, rnn_outputs_test)
```

Modelleri uygulayın:

Yeni stereo müzikten enstrümanları ayrıştırmak için, stereo ses dosyasını yükleyebilir, ses sinyalini büyüklük spektrogramına önişlemleyebilir ve CNN ve RNN modellerine girerek her bireysel enstrüman izinin tahmin edilen büyüklük spektrogramlarını elde edebilirsiniz. Son olarak, büyüklük spektrogramlarını zaman alanına geri dönüştürebilir ve her bireysel enstrüman izini ayrı ses dosyaları olarak kaydedebilirsiniz. Burada bir örnek kod parçacığı var:

```
# load stereo audio file
mix, sr = librosa.load('path/to/new/stereo/file.wav', sr=None, mono=False)

# convert audio signal to magnitude spectrogram
mix_spec = np.abs(librosa.stft(mix))

# use CNN model to predict magnitude spectrogram of individual instrument tracks
cnn_pred = cnn_model.predict(mix_spec[np.newaxis, ..., np.newaxis])[0]

# use RNN model to refine prediction of individual instrument tracks
rnn_pred = rnn_model.predict(cnn_pred[np.newaxis, ...])[0]

# convert magnitude spectrograms to time domain signals
vocals_pred = librosa.istft(vocals_pred * mix / mix_spec)
drums_pred = librosa.istft(drums_pred * mix / mix_spec)
bass_pred = librosa.istft(bass_pred * mix / mix_spec)
other_pred = librosa.istft(other_pred * mix / mix_spec)

# save individual instrument tracks as separate audio files
librosa.output.write_wav('path/to/new/vocals/file.wav', vocals_pred, sr)
librosa.output.write_wav('path/to/new/drums/file.wav', drums_pred, sr)
librosa.output.write_wav('path/to/new/bass/file.wav', bass_pred, sr)
librosa.output.write_wav('path/to/new/other/file.wav', other_pred, sr)
```

Şimdi, CNN ve RNN modellerinizi oluşturmanın zamanı geldi. CNN modeli, karışık ses sinyaline ait büyüklük spektrogramını girdi olarak alacak ve her bir enstrüman parçası için büyüklük spektrogramını tahmin edecek.

RNN modeli, karışık ses sinyaline ait büyüklük spektrogramını ve her enstrüman parçası için tahmin edilen büyüklük spektrogramını girdi olarak alarak bu tahminleri düzelterek her bir enstrüman parçası için büyüklük spektrogramını tahmin edecek. Modellerinizi oluşturduktan sonra, girdi ve çıktı verilerinizi kullanarak onları eğitmeye başlayabilirsiniz.

CNN modeli için Adam optimizatörü ve ortalama kare hatası kayıp fonksiyonu, RNN modeli için kategorik çapraz entropi kayıp fonksiyonu kullanabilirsiniz. Modellerinizi eğittikten sonra, ortalama kare hata, sinyal-gürültü oranı ve kaynak-distorsiyon oranı gibi metrikleri kullanarak performanslarını değerlendirmenin zamanı geldi.

Ayrıca, tahmin edilen spektrogramları gözlemleyerek modellerin tek tek enstrüman parçalarını nasıl ayırdıklarını görebilirsiniz. Bu nedenle, CNN ve RNN kullanarak stereo müzikten enstrümanları ayrıştırmak eğlenceli ve zorlu bir görevdir ve heyecan verici sonuçlara yol açabilir! Müzisyen veya sadece müzik seven biriyseniz, bu teknik sizin favori parçalarınızdan belirli enstrümanları çıkarmaya ve benzersiz remixler oluşturmaya yardımcı olacaktır. Ve sinir ağı teknolojisinin sürekli gelişiminin devam etmesiyle, gelecekte neler keşfedeceğiz bilinmez!

Çeviri

Taha Semih Açıkgöz

// WeAreTheArtMakers.com