MUSIC RESAMPLER

Team 03

Yesin Soufi

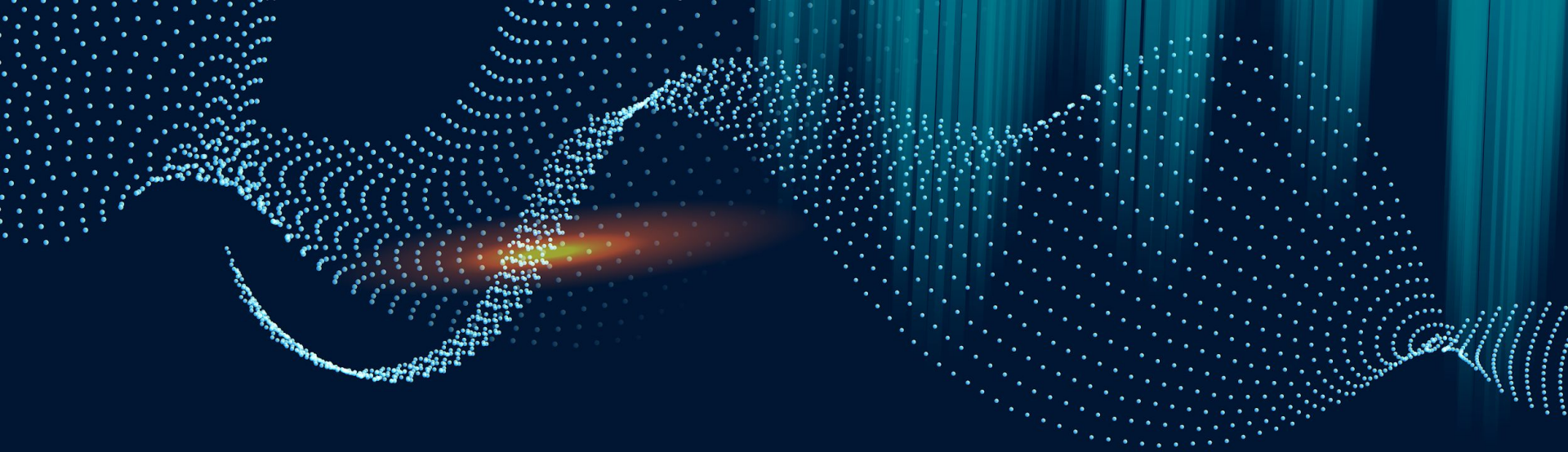
Osman Kaplan

Jakob Schaal

Niklas Öxle

Elena Müller

Sascha Lehmann



01 Challenge 1

Create Samples

Create Samples

- Es wurden Songs ohne Gesang genutzt
  - Ca. 30 Songs aus einem Genre
  - (https://drive.google.com/drive/folders/1y46vfqsB-ZrLwG3UFxaXqDNHz14qxKXh)

- Die Songs wurden zunächst in gleich lange Samples geschnitten

- Darauf aufbauend wurden verschiedene Sample Trainings Datensätze erzeugt

Create Samples (3 Python files)

createSamples.py

https://bit.ly/3sKHtMz

createDataSet.py

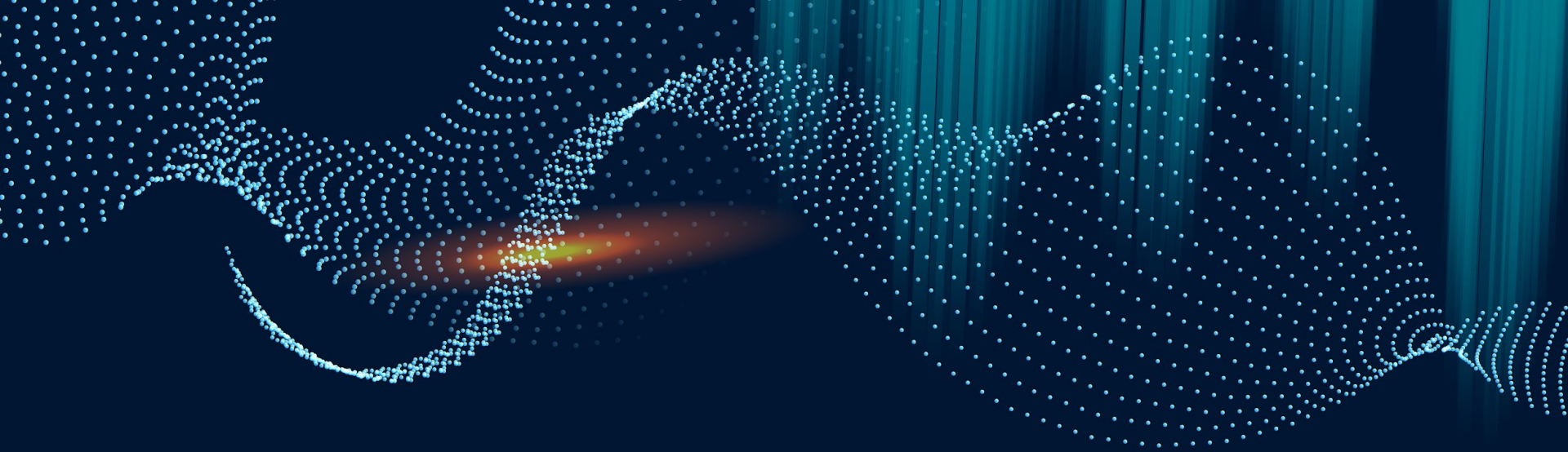
```
# %%
import createSamples
import feature_extraction
import cluster_dataset
import pandas as pd

# %%
# variables
toCutAudioPath = '../AudioData/AudioData.wav'
sampleSavePath = '../AudioData/AudioDataSamples/'
sampleLength = 4
cluster = 200

# %%
# create samples from cutting one long track
df_dataset_samples = createSamples.createSamples(myAudioPath=toCutAudioPath, savePath=sampleSavePath, sampleLength=sampleLength)
df_dataset_samples.to_csv('../dataset_csv/dataset_samples_' + str(sampleLength) + '_seconds.csv', index=False)
df_dataset_samples

# %%
# extract features from AudioData.wav samples
df_load_samples = pd.read_csv('../dataset_csv/dataset_samples_' + str(sampleLength) + '_seconds.csv')
df_dataset_features = feature_extraction.extract_features(df_load_samples)
df_dataset_features.to_csv('../dataset_csv/dataset_features_' + str(sampleLength) + '_seconds.csv', index=False)
df_dataset_features

# %%
# cluster features and append label column
df_load_features = pd.read_csv('../dataset_csv/dataset_features_' + str(sampleLength) + '_seconds.csv')
df_dataset_labels = cluster_dataset.cluster_data(df_load_features, cluster = cluster)
df_dataset_labels.to_csv('../dataset_csv/dataset_labels_' + str(sampleLength) + '_seconds_' + str(cluster) + '_cluster.csv', index=False)
df_dataset_labels

# %%
```

https://bit.ly/3KSxwCF

createDataSet.py

```
import pandas as pd
import numpy as np
from pathlib import Path
import librosa
import librosa.display


#expecting dataframe with unlabeled samples dataset
#returns dataframe with added audio features
def extract_features(df_raw_dataset):
    features = ['filePath', 'rmse', 'chroma_stft', 'spec_cent', 'spec_bw', 'rolloff', 'zcr', 'mfcc']
    features_data = []
    count = 1
    for audioFile in df_raw_dataset['filePath']:
        print('sample-file nr.: ' + str(count))
        features_data.append(get_features(audioFile))
        count = count + 1

    df_song_features = pd.DataFrame(features_data, columns=features)
    df_features_dataset = df_raw_dataset.merge(df_song_features, on='filePath')

    del df_raw_dataset
    del df_song_features

    return df_features_dataset

#load audio-file
#calculate features
#return feature array
def get_features(filePath):
    features_data = []
    y, sr = librosa.load(filePath)

    rmse = librosa.feature.rms(y=y)
    chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
    spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
    spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
    rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
    zcr = librosa.feature.zero_crossing_rate(y)
    mfcc = librosa.feature.mfcc(y=y, sr=sr)

    return [filePath, np.mean(rmse),np.mean(chroma_stft), np.mean(spec_cent),np.mean(spec_bw),np.mean(rolloff), np.mean(zcr), np.mean(mfcc)]
```

https://bit.ly/3KSxwCF



02 Challenge 2

Rebuild song from samples

Convolutional Neural Network

buildModel.py

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense,Dropout,Activation,Flatten
from tensorflow.keras.optimizers import Adam

def firstModel(num_labels):
    model=Sequential()
    ###first layer
    model.add(Dense(100,input_shape=(176400,)))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    ###second layer
    model.add(Dense(200))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    ###third layer
    model.add(Dense(100))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    ###fourth layer
    model.add(Dense(200))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    ###fifth layer
    model.add(Dense(100))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    ###final layer
    model.add(Dense(num_labels))
    model.add(Activation('softmax'))

    model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

    return model

def cnnModel():
    model = Sequential()
    model.add(Conv1D(64, 3, activation='relu', input_shape=(176400,1)))
    model.add(MaxPooling1D(2))
    model.add(Conv1D(64, 3, activation='relu'))
    model.add(MaxPooling1D(2))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(24, activation='softmax'))

    model.compile(Adam(lr=.0001), loss='categorical_crossentropy', metrics=['accuracy'])

    return model
```

https://bit.ly/3M38OkG

tensorflow_classification.py

```python
#%%
# Label Parameter
N_CLASSES = 25 # CHANGE HERE, total number of classes

#Files containing the path to images and the labels [path/to/images label]
train_file = 'data/csv/train_Samples.csv'
val_file = 'data/csv/vali_Samples.csv'

#Lists where to store the paths and labels will be stored
files = []
labels = []

df_test = pd.read_csv(train_file, index_col=0)
df_test
#%%
#load waveform from samples with filePath
extracted_waveform=[]
for index_num,row in df_test.iterrows():
    file_name = row['filePath']
    file_name = file_name.replace('Sascha', 'sasch')
    class_label = row["label"]
    data, sr = librosa.load(file_name, mono=True)
    extracted_waveform.append([data,class_label])

extracted_waveform

#%%
#create df from extracted waveform and label
extracted_df=pd.DataFrame(extracted_waveform,columns=['waveform','class'])
extracted_df.head(10)
extracted_df = extracted_df.drop(extracted_df[extracted_df['waveform'].map(len) < 176400].index)

extracted_df

#%%
#split into waveform and label
X=np.array(extracted_df['waveform'].tolist(), dtype='float32')
#X=np.array(X).astype("float32")
y=np.array(extracted_df['class'].tolist())

```

https://bit.ly/3KZ4Msf

Vorgehensweise Challenge 2

- Setzen von Label Parametern
- Für die Klassifikation haben wir 2 csv Dateien verwendet (train_file und val_file)
- Im Trainingsdatensatz wurden die Samples mit einer nach der Reihenfolge gesetzten ID des Songs gelabelt
- Anschließend wurde ein neuer Song genommen der nicht im Trainingsdatensatz enthalten war und wird in 8 sek Stücke geschnitten und sollte für jedes Sample das zugehörige Label prognostizieren
- Im Letzten Schritt werden die Samples nach den prognostizierten Samples sortiert und exportiert

Ergebnisse

Original Song

Trained Song