

# MUSIC RESAMPLER

---

Team 03

Yesin Soufi

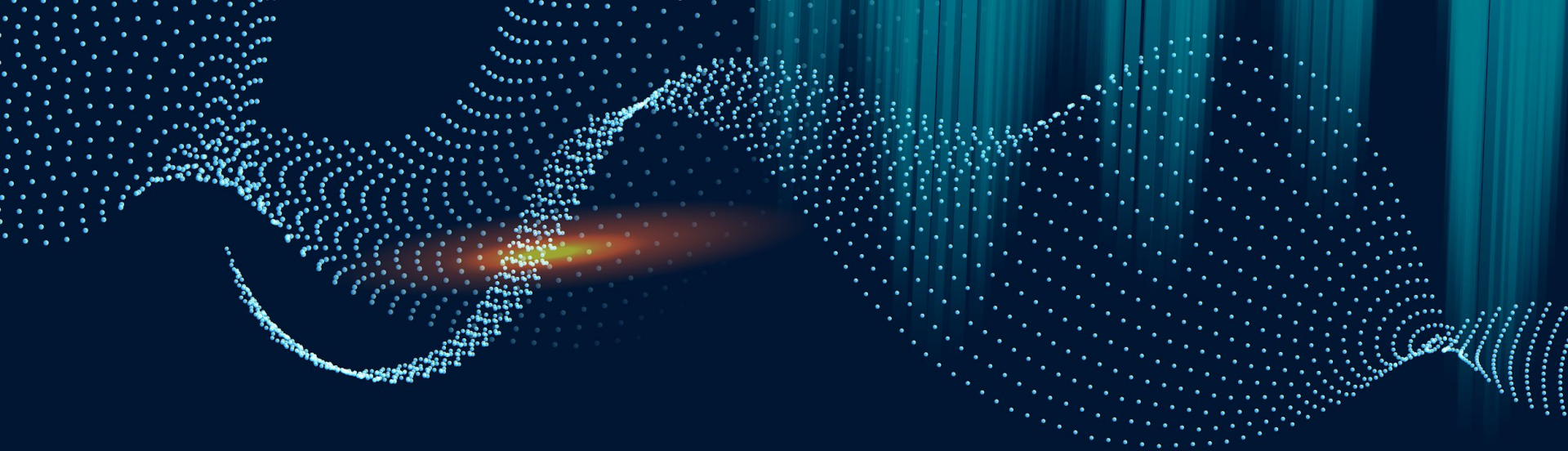
Osman Kaplan

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

createDataSet.py

createDataSet.py

```
1 # 50
2 import numpy as np
3 import pandas as pd
4 import os
5
6 from glob import glob
7 import IPython.display as ipd
8
9 from pydub import AudioSegment
10 from pydub.utils import make_chunks
11
12 import shutil
13 import random
14 from pathlib import Path
15
16 from pyperclip import copy
17 # 50
18
19 def cutSamples(myAudioPath, savePath, sampleLength, overlap = 0):
20     segment = AudioSegment.from_file(myAudioPath)
21     chunk_size = (sampleLength*1000) # pydub calculates in millisec
22     for chunk_length_ms in chunk_size:
23         chunks = make_chunks(segment, chunk_length_ms) # Make chunks of one sec
24         for i, chunk in enumerate(chunks):
25             chunk_name = str(i+1) + ".wav"
26             chunk.export(savePath + chunk_name, format='wav')
27     return print("Samples export successful")
28
29 def createSampleID(audioPaths):
30     data = []
31     for file in sorted(Path(audioPaths).glob("*.wav")):
32         data.append([os.path.basename(file), file])
33    
34     df_dataset = pd.DataFrame(data, columns = ["audio name", "filePath"])
35     df_dataset["ID"] = df_dataset.index
36     df_dataset = df_dataset[["ID", "audio name", "filePath"]]
37     return df_dataset
38
39 def createSampleID(myAudioPath, savePath, sampleLength, overlap = 0):
40     cutSamples(myAudioPath=myAudioPath, savePath=savePath, sampleLength=sampleLength)
41     df_dataset=createSampleID(audioPaths=savePath)
42     df_dataset=sort_Dataframe(df_dataset)
43     return df_dataset
44
45 def sort_Dataframe(df_dataset):
46     df_to_sort = df_dataset[["audio name", "filePath"]].copy()
47     df_to_sort["audio name"] = df_to_sort["audio name"].str.extract("(.*)")
48     df_to_sort["audio name"] = df_to_sort["audio name"].astype(int)
49     df_to_sort.sort_values(by="audio name", inplace=True)
50     df_to_sort["audio name"] = df_to_sort["audio name"].astype(str) + ".wav"
51     df_dataset = df_dataset.copy()
52     df_dataset["audio name", :]
53     df_dataset = df_dataset.sort("filePath", 1)
54     df_to_sort.reset_index(inplace=True)
55     df_dataset = df_dataset.join(df_to_sort)
56     df_sorted = df_dataset.drop("index", 1)
57
58     return df_sorted
```

```
1 # 300
2 import createSamples
3 import feature_extraction
4 import cluster_dataset
5 import pandas as pd
6
7 # 300
8 # variables
9 toCutAudioPath = '../AudioData/AudioData.wav'
10 sampleSavePath = '../AudioData/AudioDataSamples/'
11 sampleLength = 4
12 cluster = 200
13
14 # 300
15 # create samples from cutting one long track
16 df_dataset_samples = createSamples.createSamples(myAudioPath=toCutAudioPath, savePath=sampleSavePath, sampleLength=sampleLength)
17 df_dataset_samples.to_csv('../dataset_csv/dataset_samples_' + str(sampleLength) + '_seconds.csv', index=False)
18 df_dataset_samples
19
20 # 300
21 # extract features from AudioData.wav samples
22 df_load_samples = pd.read_csv('../dataset_csv/dataset_samples_' + str(sampleLength) + '_seconds.csv')
23 df_load_features = feature_extraction.extract_features(df_load_samples)
24 df_dataset_features.to_csv('../dataset_csv/dataset_features_' + str(sampleLength) + '_seconds.csv', index=False)
25 df_dataset_features
26
27 # 300
28 # cluster features and append label column
29 df_load_features = pd.read_csv('../dataset_csv/dataset_features_' + str(sampleLength) + '_seconds.csv')
30 df_dataset_labels = cluster_dataset.cluster_data(df_load_features, cluster = cluster)
31 df_dataset_labels.to_csv('../dataset_csv/dataset_labels_' + str(sampleLength) + '_seconds_' + str(cluster) + '_cluster.csv', index=False)
32 df_dataset_labels
33
34 # 300
```

```
1 import pandas as pd
2 import numpy as np
3 from pathlib import Path
4 import librosa
5 import librosa.display
6
7
8 # Respecting dataframe with unlabeled samples dataset
9 # Returns dataframe with added audio features
10 def extract_features(df_raw_dataset):
11     features = ["filePath", "rms", "chroma_stft", "spec_cent", "spec_bw", "rolloff", "zcr", "mfcc"]
12     features_data = []
13     count = 1
14     for audioFile in df_raw_dataset["filePath"]:
15         print("sample-file nr.: " + str(count))
16         features_data.append(get_features(audioFile))
17     count = count + 1
18
19 df_sung_features = pd.DataFrame(features_data, columns=features)
20 df_features_dataset = df_raw_dataset.merge(df_sung_features, on="filePath")
21
22 del df_raw_dataset
23 del df_sung_features
24
25 return df_features_dataset
26
27 # Read audio-file
28 # Calculate features
29 # Returns feature array
30 def get_features(filePath):
31     features_data = []
32     y, sr = librosa.load(filePath)
33
34     rms = librosa.feature.rms(y=y)
35     chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
36     spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
37     spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
38     rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
39     zcr = librosa.feature.zero_crossing_rate(y)
40     mfcc = librosa.feature.mfcc(y=y, sr=sr)
41
42     return [filePath, np.mean(rms), 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>

<https://bit.ly/3sKHtMz>

<https://bit.ly/3KSxwCF>





02

## Challenge 2

Rebuild song from samples

# Convolutional Neural Network

## buildModel.py

```
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
```

## tensorflow\_classification.py

```
22  ###
23  # Label Parameter
24  N_CLASSES = 25 # CHANGE HERE, total number of classes
25
26  #files containing the path to images and the labels [path/to/images label]
27  train_file = 'data/csv/train_Samples.csv'
28  val_file = 'data/csv/vali_Samples.csv'
29
30  #Lists where to store the paths and labels will be stored
31  files = []
32  labels = []
33
34  df_test = pd.read_csv(train_file, index_col=0)
35  df_test
36  ###
37  #load waveform from samples with filePath
38  extracted_waveform=[]
39  for index_num,row in df_test.iterrows():
40      file_name = row['filePath']
41      file_name = file_name.replace('Sascha', 'sasch')
42      class_label = row["label"]
43      data, sr = librosa.load(file_name, mono=True)
44      extracted_waveform.append([data,class_label])
45
46  extracted_waveform
47
48  ###
49  #create df from extracted waveform and label
50  extracted_df=pd.DataFrame(extracted_waveform,columns=['waveform','class'])
51  extracted_df.head(10)
52  extracted_df = extracted_df.drop(extracted_df[extracted_df['waveform'].map(len) < 176400].index)
53
54  extracted_df
55
56  ###
57  #split into waveform and label
58  X=np.array(extracted_df['waveform'].tolist(), dtype="float32")
59  #X=np.array(X).astype("float32")
60  y=np.array(extracted_df['class'].tolist())
61
```

<https://bit.ly/3M38OkG>

<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**

---

