Öğr. Gör. Dr. Ahmet ELBİR BLM3620 Sayısal İşaret İşleme

**Son Teslim tarihi:** **31.05.2023 23.59**

Proje

Oğuzhan Topaloğlu

Ç19052025 – Grup 1

*Bilgisayar Mühendisliği Bölümü,*

*Elektrik-Elektronik Fakültesi,*

*Yıldız Teknik Üniversitesi*

****

Istanbul, 2023

|  |
| --- |
| **İÇERİK** |
| 1. STFT İşlemi |
| 2. Tablo Üretimi |
| 3. Ekran Görüntüleri |
|  |
|  |
|  |
|  |
|  |

**1. STFT İşlemi**

Projede main kısmında çalıştırılabilecek 2 adet fonksiyon bulunmaktadır:

if \_\_name\_\_ == '\_\_main\_\_':

    # Hangi main'in çalıştırılması isteniyorsa öbürü yoruma alınmalı

    # main\_create\_csv\_files()

    main\_knn()

Bunlardan main\_create\_csv\_files fonksiyonu bütün WAV dosyalarını okuyarak projede istenen 6 adet CSV dosyasını üretmektedir. Main\_knn fonksiyonu da bu 6 CSV dosyasını okuyarak yine projede istenen tabloları oluşturmaktadır. (KNN algoritmasını sklearn’den kullanarak).

CSV üreten main fonksiyonu aşağıdaki gibidir:

def main\_create\_csv\_files():

    create\_specific\_csv("Blackman", np.blackman, True)

    create\_specific\_csv("Hamming", np.hamming, True)

    create\_specific\_csv("Hanning", np.hanning, True)

    create\_specific\_csv("Blackman", np.blackman, False)

    create\_specific\_csv("Hamming", np.hamming, False)

    create\_specific\_csv("Hanning", np.hanning, False)

Görüldüğü gibi, kullanılacak window türünün ismini, fonksiyonunu ve True/False ile test/train olup olmadığını belli ederek başka bir fonksiyonu çağırmaktadır. Bu fonksiyon da aşağıdaki gibidir:

def create\_specific\_csv(window\_name, window\_func, is\_train):

    train\_or\_test = "Train" if is\_train else "Test"

    csv\_name = f"{window\_name}\_{train\_or\_test}.csv"

    df\_list = []

    for music\_type in ["classical", "country", "jazz", "metal", "reggae"]:

        folder\_path = os.path.join("Data", music\_type, train\_or\_test.lower())

        for music\_file\_path in os.listdir(folder\_path):

            music\_file\_path = os.path.join(folder\_path, music\_file\_path)

            stft\_result = my\_stft(music\_file\_path, window\_func)

            features\_array = get\_features\_vector(stft\_result)

            features\_array.append(music\_type)

            df\_list.append(features\_array)

    result\_df = pd.DataFrame(df\_list)

    result\_df.to\_csv(csv\_name, index=False)

Fonksiyon öncelikle oluşturulacak CSV dosyasının ismini belirlemekte ve sonrasında iki for döngüsü ile seçilen 5 müzik türünde her müzik dosyasını alarak my\_stft ile STFT hesaplamakta ve ardından get\_features\_vector ile de özellik çıkarımı yapmaktadır.

STFT işlemleri aşağıdaki fonksiyonda yapılmaktadır:

def my\_stft(file\_path, window\_func):

    # data = işlenecek sinyali içeren bir numpy dizisi

    # fs = verinin örnekleme frekansı olan skalar

    data, fs = read\_wav\_file(file\_path)

    fft\_size = 4096 # toplantıda 4000-5000 arası bir şey uygundur denildi

    overlap\_fac = 0.2  # raporda hep %20 alınmalı denildi

    hop\_size = np.int32(np.floor(fft\_size \* (1-overlap\_fac)))

    pad\_end\_size = fft\_size  # son segment

    total\_segments = np.int32(np.ceil(len(data) / np.float32(hop\_size)))

    window = window\_func(fft\_size)

    inner\_pad = np.zeros(fft\_size) # her segmentin boyutunu iki katına çıkarmak için kullanılacak sıfırlar

    proc = np.concatenate((data, np.zeros(pad\_end\_size)))

    result = np.empty((total\_segments, fft\_size), dtype=np.float32)

    for i in range(total\_segments):

        current\_hop = hop\_size \* i

        segment = proc[current\_hop:current\_hop+fft\_size]

        windowed = segment \* window

        padded = np.append(windowed, inner\_pad)

        spectrum = my\_fft(padded) / fft\_size              # benim fonksiyonumla FFT al

        autopower = np.abs(spectrum \* np.conj(spectrum))

        result[i, :] = autopower[:fft\_size]

    result = 20 \* np.log10(result)        # dB'ye ölçeklendir

    result = np.clip(result, -40, 200)    # değerleri kırp, linkte neden yaptığını açıklamış gerek yok aslında

    return result

Bu fonksiyon şu kaynaktan alınmış ve proje boyunca değiştirilmiştir:  
<https://kevinsprojects.wordpress.com/2014/12/13/short-time-fourier-transform-using-python-and-numpy/>

Fonksiyon içinde öncelikle müzik dosyası bir numpy array’ine aşağıdaki fonksiyon ile okunmuştur:

# Verilen path'i bir numpy dizisine okur

def read\_wav\_file(file\_path):

    wav = wave.open(file\_path, 'r')

    data = wav.readframes(-1)

    data = np.frombuffer(data, dtype=np.int16)

    fs = wav.getframerate()

    wav.close()

    return data, fs

Ardından toplantıda denildiği gibi fft\_size, overlap\_fac belirlenmiş ve SFTF algoritması kodlanmıştır. Algoritmada hazır FFT fonksiyonu kullanılmamış ve aşağıdaki reküsif ve benim tarafımdan yazılmış FFT fonksiyonu kullanılmıştır:

# Benim FFT fonksiyonum

def my\_fft(data):

    N = len(data)

    if N <= 1:

        return data

    even = my\_fft(data[0::2])  # Çift indislere FFT uygula

    odd = my\_fft(data[1::2])   # Tek indislere FFT uygula

    T = np.exp(-2j \* np.pi \* np.arange(N) / N)  # Dönüşüm matrisi

    return np.concatenate([even + T[:N // 2] \* odd, even + T[N // 2:] \* odd])

Bu SFTF sonucu hesaplanan result ile de aşağıdaki fonksiyon sayesinde projede istenilen üç özelliğin ortalaması, medyanı ve standart sapması hesaplanmış ve bir liste ile dönülmüştür:

# STFT sonucundan özellik çıkarımı

def get\_features\_vector(stft\_result):

    # v1: frequency\_power

    v1 = np.mean(stft\_result, axis=1)

    # v2: frequency\_amplitude

    v2 = np.mean(np.abs(stft\_result), axis=1)

    # v3: weighted\_average\_frequency

    frequencies = np.fft.fftfreq(stft\_result.shape[1])

    v3 = np.sum(np.abs(stft\_result) \* frequencies, axis=1) / np.sum(np.abs(stft\_result), axis=1)

    return [\

        np.mean(v1), np.median(v1), np.std(v1), \

        np.mean(v2), np.median(v2), np.std(v2), \

        np.mean(v3), np.median(v3), np.std(v3)]

**2. Tablo Üretimi**

KNN algoritması ve tablo üretimi öbür main fonksiyonunda yapılmaktadır:

def main\_knn():

    create\_table(k=1, window\_type="Blackman")

    create\_table(k=3, window\_type="Blackman")

    create\_table(k=5, window\_type="Blackman")

    create\_table(k=1, window\_type="Hamming")

    create\_table(k=3, window\_type="Hamming")

    create\_table(k=5, window\_type="Hamming")

    create\_table(k=1, window\_type="Hanning")

    create\_table(k=3, window\_type="Hanning")

    create\_table(k=5, window\_type="Hanning")

Görüldüğü gibi bu main fonksiyonu içinde create\_table fonksiyonunu çağırmakta ve kullanılan window türü ile k sayısını parametre olarak aktarmaktadır. Bu fonksiyon da aşağıdaki şekildedir:

def create\_table(k, window\_type):

    test\_path = os.path.join("Dataset", window\_type+"\_Test.csv")

    train\_path = os.path.join("Dataset", window\_type+"\_Train.csv")

    train\_df = pd.read\_csv(train\_path)

    train\_df.columns = ["freqPowerMean", "freqPowerMed", "freqPowerStd", "freqAmpMean", \

              "freqAmpMed", "freqAmpStd", "weiFreqMean", "weiFreqMed", "weiFreqStd", "type"]

    # Train the knn

    X = train\_df.drop('type', axis=1)

    y = train\_df['type']

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

    knn = KNeighborsClassifier(n\_neighbors=k)

    knn.fit(X\_train, y\_train)

    # Test the knn

    test\_df = pd.read\_csv(test\_path)

    test\_df.columns = ["freqPowerMean", "freqPowerMed", "freqPowerStd", "freqAmpMean", \

              "freqAmpMed", "freqAmpStd", "weiFreqMean", "weiFreqMed", "weiFreqStd", "type"]

    X = test\_df.drop('type', axis=1)

    y = test\_df['type']

    y\_pred = knn.predict(X)

    count = len(y\_pred)

    correct = 0

    for i in range(len(y\_pred)):

        if y\_pred[i] == y[i]:

            correct += 1

    table\_data = []

    for i in range(len(y\_pred)):

        table\_data.append([y[i], y\_pred[i]])

    fig, ax = plt.subplots()

    ax.axis('off')

    table = ax.table(cellText=table\_data, colLabels=['Actual', 'Predicted'],

                     cellLoc='center', loc='center', edges='horizontal')

    table.auto\_set\_font\_size(False)

    table.set\_fontsize(12)

    table.scale(1.2, 1.2)

    table\_title = f"{window\_type}, k={k}, Correctness: {correct}/{count}"

    ax.set\_title(table\_title, fontweight='bold', fontsize=14, y=2)

    # Save the figure as an image

    if not os.path.exists("Tables"):

        os.mkdir("Tables")

    plt.savefig(f"Tables/{window\_type}\_{k}.png", bbox\_inches='tight', dpi=300)

Fonksiyonda öncelikle verilen parametrelere göre train ve test CSV dosyalarının path’i bulunmaktadır. Ardından train dosyası okunarak bir KNN algoritması verilen k parametresine göre oluşturulmakta ve train edilmektedir. Model oluşturulduktan sonra da test dosyası okunmakta ve bilinen type değeri silinerek modele tür tahmini yaptırılmaktadır. Ardından da yapılan bu tahminler tablo şeklinde Tables klasöründe png dosyası olarak oluşturulmaktadır. Bu görseller bir sonraki bölümde bulunmaktadır.

**3. Ekran Görüntüleri**















