Ses İşleme - Rap Türlerinin Sınıflandırılması

Cengizhan Durmuş
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
201307087
201307087@kocaeli.edu.tr

I. GIRIŞ

Ses sınıflaması, makine öğrenimi ve uygulamalı bilgisayar bilimleri için en önemli araştırma konularından biridir. Bu çalışma sesleri kullanarak rap türlerini sınıflandırma yöntemi sunmaktadır.

Herkese açık sesler kullanılarak, boombap, jazz rap, old school, funky ve trap türlerine ait sesler YouTube üzerinde indirildi. Bu dosyalar farklı tarihlerde elde edilip indirilerek türdeki her ses dosyası için 15 saniyelik kesitler üretilmiştir. Bu şekilde, veri kümemizde toplam 12.198 dosya elde edildi.

Bu çalışmada rap türlerinin sınıflandırması için 1 adet veri kümesi kullanılmıştır. Aşağıdaki tabloda veri kümesindeki toplam ses dosyası sayısı Boom bap, Old school, Jazz rap, Funky ve Trap türlerin de bulunan veri sayısı gösterilmiştir.

- 1 adet Yapay Sinir Ağı modeli (YSA)
- 1 adet Derin Öğrenme modeli (CNN)
- 1 adet Makine Öğrenme modeli (DT)

kullanılmıştır.

Veri Kümesi

Sınıf	Dosya Sayısı
Boom Bap	2.497
Funky	2.377
Jazz Rap	2.439
Old School	2.137
Trap	2.748
Toplam	**12.198**

II. KULLANILAN ÖĞRENME MODELLERI

A. Yapay Sinir Ağı

Yapay Sinir Ağları (YSA), günümüzde Derin Öğrenme (Deep Learning) olarak geçen makine öğrenimi yaklaşımının temelini oluşturmakta.

Yapay sinir ağı (YSA), insan beyninin bilgiyi analiz etme ve işleme şeklini simüle etmek için tasarlanmış bir bilgi işlem sisteminin parçasıdır diyebiliriz

```
from sklearn.preprocessing import La
belEncoder, StandardScaler
import keras
import pandas as pd
import numpy as np
from sklearn.model_selection import
train_test_split
from keras.models import Sequential
from keras.layers import Dense
```

Öğrenme modelini uygulamak için öncelikle yukarıdaki kütüphaneleri import ediyoruz. Alt satırda ise ön işleme yapılmış csv dosyamızı çekiyoruz.

```
data = pd.read_csv('dataAudio.csv')

data.head()
```

Etiketleri kodluyoruz.

```
data = data.drop(['filename'],axis=1
)
genre_list = data.iloc[:, -1]
encoder = LabelEncoder()
```

Özellik sütunlarını ölçeklendirme.

```
y = encoder.fit_transform(genre_list
)
scaler = StandardScaler()
```

Verileri eğitim ve Test setine bölme işlemi.

```
X = scaler.fit_transform(np.array(da
ta.iloc[:, :-1], dtype = float))
X_train, X_test, y_train, y_test = t
rain_test_split(X, y, test_size=0.3)
```

Sıradaki aşama model oluşturma. Kullandığımız parametreler input_dim = özellikler, units = yapay sinir hücreleri, init = yapay sinir hücresi içerisindeki ağırlıkları rastgele yerine belirli algoritmaya göre verir.

```
model = Sequential()

model.add(Dense(256, activation='relu'
, input_shape=(X_train.shape[1],)))

model.add(Dense(128, activation='relu'
))

model.add(Dense(64, activation='sigmoi
d'))

model.add(Dense(10, activation='sigmoi
d'))

model.compile(optimizer='adam',loss='s
parse_categorical_crossentropy',metric
s=['accuracy'])
```

sparse_categorical_crossentropy = Etiketler ve tahminler arasındaki çapraz entropi kaybını hesaplar.

Modeli uygulama kısmında kullandığımız parametreler ise epochs = ağırlıkları yeniden düzenleyerek çözüme en yakın değeri buluncaya kadar yapılacak deneme sayısı, batch_size = tek seferde alınacak veri sayısı, verbose = hatanın gösterilmemesi, validation_split = doğrulama için eğitim verilerinin bir kısmını otomatik olarak ayırmaya izin verir.

```
history = model.fit(X_train,y_train,
validation_split=0.33,epochs=200,bat
ch_size=100)
```

Model doğruluğunu hesaplamak ve yazdırmak için de alttaki kod satırını kullanıyoruz.

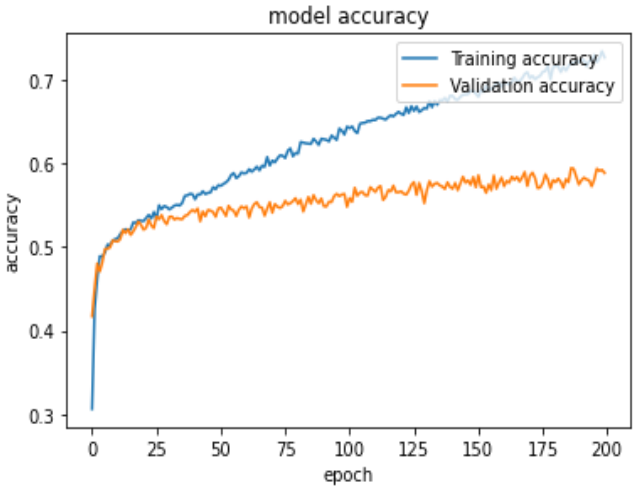
```
test_loss, test_acc = model.evaluate
(X_train,y_train)
print("The test loss is: ", test_los
s)
print("The best accuracy is: %", tes
t_acc*100)
```

Modelimizin accuracy değeri = 0.8596

Uyguladığımız Yapay Sinir Ağı makine öğrenimi modelinin performansının görselleştirilmesi, modelden dökülen verileri anlamlandırmanın ve Makine Öğrenimi modelini etkileyen parametreler veya hiperparametreler üzerinde yapılması gereken değişiklikler hakkında bilinçli bir karar vermenin kolay yoludur.

Matplotlib kütüphanesini kullanarak verileri görselleştirme işlemini yapacağız. Accuracy geçmişi için ;

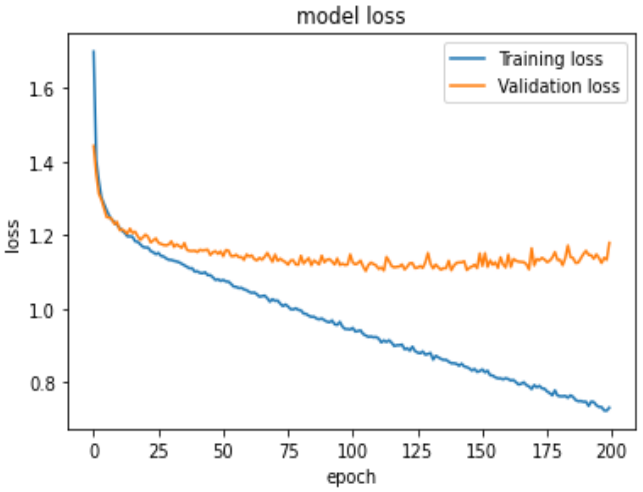
```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy']
)
plt.plot(history.history['val_accura
cy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Training accuracy', 'Va
lidation accuracy'], loc='upper righ
t')
plt.show()
```



Şekil 1: YSA accuracy geçmişi

Loss geçmişi için de ;

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss']
)
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Training loss', 'Valida
tion loss'], loc='upper right')
plt.show()
```



Şekil 2: YSA loss geçmişi

B. CNN (Convolutional Neural Network)

Evrişimsel sinir ağı", sinir ağının evrişim adı verilen bir matematik işlemi kullandığını gösterir. Konvolüsyon (evrişim) özel bir doğrusal (lineer) işlem türüdür. Evrişimli sinir ağları, katmanlarından en az birinde genel matris çarpımı yerine evrişimi kullanan basit sinir ağlarıdır.

```
from sklearn.preprocessing import La
belEncoder, StandardScaler
import keras
import pandas as pd
import numpy as np
from sklearn.model_selection import
train_test_split
import tensorflow as tf
import matplotlib.pyplot as plt
```

Öğrenme modelini uygulamak için öncelikle yukarıdaki kütüphaneleri import ediyoruz. Alt satırda ise ön işleme yapılmış csv dosyamızı çekiyoruz.

```
df = pd.read_csv('dataAudio.csv')
df.head()
```

Veri setimizdeki satır – sütün sayısını ve sütünlardaki veri tiplerini alttaki kod satırları ile listeliyoruz.

```
df.shape
df.dtypes
```

Metni doğrudan eğitim için kullanamayız. Bu etiketleri sklearn.preprocessing'in LabelEncoder() işleviyle kodladık. Alt satırda ise kategorik metin verilerini modelin anlayabileceği sayısal verilere dönüştürme işlemini yapıyoruz.

```
class_list = df.iloc[:,-1]
encoder = LabelEncoder()
y = encoder.fit_transform(class_list
)
print("y: ", y)
```

input_parameters değişkeni dosya adı dışındaki tüm diğer sütunlar sinir ağının giriş parametreleridir. Eğitim için herhangi bir bilgi sağlamayan ilk sütunu (dosya adı) ve çıktıya karşılık gelen son sütunu kaldırdık.

```
input_parameters = df.iloc[:, 1:7]
scaler = StandardScaler()
X = scaler.fit_transform(np.array(in
put_parameters))
print("X:", X)
```

Verileri eğitim ve Test setine bölme işlemi.

```
X_train, X_val, y_train, y_val = tra
in_test_split(X, y, test_size = 0.3)
```

Sıradaki aşama model oluşturma. Kullandığımız parametreler input_dim = özellikler, units = yapay sinir hücreleri, init = yapay sinir hücresi içerisindeki ağırlıkları rastgele yerine belirli algoritmaya göre verir.

```
model = tf.keras.models.Sequential([
```

```
    tf.keras.layers.Dense(512, activ
ation = 'relu', input_shape = (X_tra
in.shape[1],)),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(256, activ
ation = 'relu'),
    keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(128, activ
ation = 'relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(64, activa
tion = 'relu'),
    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(45, activa
tion = 'softmax'),
])

print(model.summary())
```

Adam optimizer, modeli 100'den fazla eğitmek için kullanılır. Bu seçim, daha iyi sonuçlar elde etmemizi sağladığı için yapılmıştır.

Kayıp (loss), sparse_categorical_crossentropy işleviyle hesaplanır.

```
def trainModel(model,epochs, optimiz
er):
    batch_size = 128
    model.compile(optimizer = optimi
zer, loss = 'sparse_categorical_cros
sentropy', metrics = 'accuracy')
    return model.fit(X_train, y_trai
n, validation_data = (X_val, y_val),
 epochs = epochs, batch_size = batch
_size)
```

Modeli uygulama kısmında kullandığımız parametreler ise epochs = ağırlıkları yeniden düzenleyerek çözüme en yakın değeri buluncaya kadar yapılacak deneme sayısı, batch_size = tek seferde alınacak veri sayısı, verbose = hatanın gösterilmemesi, validation_split = doğrulama için eğitim verilerinin bir kısmını otomatik olarak ayırmaya izin verir.

```
model_history = trainModel(model = m
odel, epochs = 200, optimizer = 'ada
m')
```

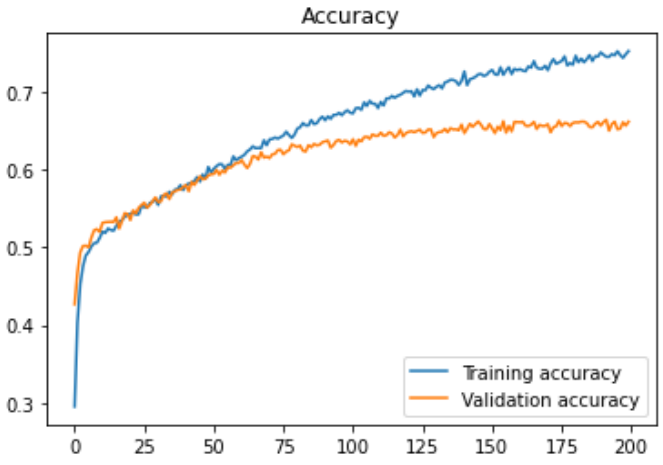
Modelimizin accuracy değeri = 0.6620

```
test_loss, test_acc = model.evaluate
(X_val, y_val, batch_size = 128)
print("The test loss is: ", test_los
s)
print("The best accuracy is: %", tes
t_acc*100)
```

Uyguladığımız CNN derin öğrenimi modelinin performansının görselleştirilmesi, modelden dökülen verileri anlamlandırmanın ve Makine Öğrenimi modelini etkileyen parametreler veya hiperparametreler üzerinde yapılması gereken değişiklikler hakkında bilinçli bir karar vermenin kolay yoludur.

Matplotlib kütüphanesini kullanarak verileri görselleştirme işlemini yapacağız. Accuracy geçmişi için ;

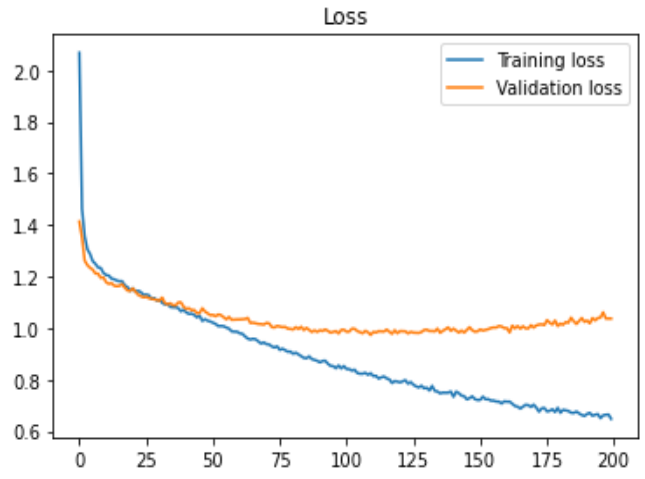
```
acc_train_curve = model_history.hist
ory["accuracy"]
acc_val_curve = model_history.histor
y["val_accuracy"]
plt.plot(acc_train_curve, label = "T
raining accuracy")
plt.plot(acc_val_curve, label = "Val
idation accuracy")
plt.legend(loc = 'lower right')
plt.title("Accuracy")
plt.show()
```



Şekil 3: CNN accuracy geçmişi

Loss geçmişi için de ;

```
loss_train_curve = model_history.his
tory["loss"]
loss_val_curve = model_history.histo
ry["val_loss"]
plt.plot(loss_train_curve, label = "
Training loss")
plt.plot(loss_val_curve, label = "Va
lidation loss")
plt.legend(loc = 'upper right')
plt.title("Loss")
plt.show()
```



Şekil 4: CNN loss geçmişi

C. Decision Tree (Karar Ağaçları)

Karar ağaçları, Sınıflandırma ve Regresyon problemlerinde kullanılan, ağaç tabanlı algoritmadan biridir. Karmaşık veri setlerinde kullanılabilir.

Bir karar ağacı, çok sayıda kayıt içeren bir veri kümesini, bir dizi karar kuralları uygulayarak daha küçük kümelere bölmek için kullanılan bir yapıdır. Yani basit karar verme adımları uygulanarak, büyük miktarlardaki kayıtları, çok küçük kayıt gruplarına bölerek kullanılan bir yapıdır.

```
import pandas as pd
from sklearn.tree import DecisionTreeC
lassifier
from sklearn.model_selection import tr
ain_test_split
from sklearn import metrics
```

Öğrenme modelini uygulamak için öncelikle yukarıdaki kütüphaneleri import ediyoruz. Alt satırda ise ön işleme yapılmış csv dosyamızı çekiyoruz.

```
df = pd.read_csv('dataAudio.csv')
col_name=['filename','chroma_frequen
cy','sp_centroid','spectral_bandwidt
h','rolloff','zero_crossing_rate','m
fcc','label']
data.columns=col_name
df.head()
```

Etiketleri kodluyoruz. Features ve label kısımlarını veri setimizden ayırıyoruz.

```
feature_cols=['chroma_frequency','sp
_centroid','spectral_bandwidth','rol
loff','zero_crossing_rate','mfcc']
X=data[feature_cols]
y=data.label
print(X)
print(y)
```

Verileri eğitim ve Test setine bölme işlemi.

```
X_train, X_test, y_train,y_test= tra
in_test_split(X,y, test_size=0.3, ra
ndom_state=1)
print(X_test)
```

DecisionTreeClassifier, bir veri kümesi üzerinde çok sınıflı sınıflandırma yapabilen bir sınıftır. DecisionTreeClassifier girdi olarak iki dizi alır.

```
dt=DecisionTreeClassifier()
dt=dt.fit(X_train,y_train)
```

Bir sınıflandırmanın doğruluğunu değerlendirmek için confusion matrix (karışıklık matrisi) hesaplıyoruz. Bu hesaplama sayesinde accuracy, precision, recall, f-measure değerlerini de hesaplayabiliriz.

```
y_pred=dt.predict(X_test)
from sklearn.metrics import confusio
n_matrix
confmat = confusion_matrix(y_test,y_
pred)
print(confmat)
```

Accuracy ile alt küme doğruluğunu hesaplama. (0.4938)

```
from sklearn.metrics import accuracy
_score

acc = accuracy_score(y_test,y_pred)
print(acc)
```

Precision (hassasiyet) hesaplama. (0.4911)

```
from sklearn.metrics import preci
sion_score

pre = precision_score(y_test, y_p
red,average='macro')
print(pre)
```

Recall (geri çağırma) hesaplama. (0.4919)

```
from sklearn.metrics import recal
l_score

rec = recall_score(y_test, y_pred
, average='macro')
print(rec)
```

F-measue, precision ve recall'un harmonik bir ortalaması olarak yorumlanabilir. (0.4913)

```
from sklearn.metrics import f1_score

fm = f1_score(y_test, y_pred, averag
e='macro')
print(fm)
```

KAYNAKLAR

[1] Confusion Matrix Python, https://intellipaat.com/blog/confusion-matrix-python/

[2] Data Visualization for Deep Learning Model Using Matplotlib, https://www.pluralsight.com/guides/data-visualization-deep-learning-model-using-matplotlib

[3] Display Deep Learning Model Training History in Keras, https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

[4] Training and evoulation with the built-in methods, https://www.tensorflow.org/guide/keras/train_and_evaluate

[5] Probabilistic Loses, https://keras.io/api/losses/probabilistic_losses/#binarycrossentropy-class

[6] Audio Data Analysis Using Deep Learning with Pyhton, https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python-part-2.html#