Uygulama 7. 3-Boyutlu Evrişimsel Sinir Ağı (ESA) ile Ses İşaretlerinin Analizi

Bu uygulama ile ses dosyalarının ESA ile analizi gerçekleştirilecektir. Kedi ve köpeğe ait sesler kullanılarak sınıflandırma işlemi gerçekleştirilecektir.

Öncelikle gerekli kütüphaneler yüklenir.

```
#gerekli kütüphanelerin yüklenmesi
import matplotlib.pyplot as plt
from collections import namedtuple
from tensorflow.python.framework import ops
import tensorflow as tf
import numpy as np
import pandas as pd
import sys
import os
import scipy.io.wavfile as sci_wav
import random
```

Kullanılacak dosyaların yolları belirlenir.

```
ROOT_DIR = "D:/CNN Ses Sınıflandırma/cats_dogs/"
CSV_PATH = "D:/CNN Ses Sınıflandırma/train_test_split.csv"
```

Ardından ses dosyalarının okunması için gerekli komutlar oluşturulur ve sistem üzerinden çağrılır.

```
def read_wav_files(wav_files):
    if not isinstance(wav_files, list):
        wav_files = [wav_files]
    return [sci_wav.read(ROOT_DIR + f)[1] for f in wav_files]

def get_trunk(_X, idx, sample_len, rand_offset=False):
    randint = np.random.randint(10000) if rand_offset is True else 0
    start_idx = (idx * sample_len + randint) % len(_X)
    end_idx = ((idx + 1) * sample_len + randint) % len(_X)
    if end_idx > start_idx:
        return _X[start_idx: end_idx]
    else:
        return np.concatenate((_X[start_idx:], _X[:end_idx]))

def get_augmented_trunk(_X, idx, sample_len, added_samples=0):
    X = get_trunk(_X, idx, sample_len)

    for _ in range(added_samples):
        ridx = np.random.randint(len(_X))
        X = X + get_trunk(_X, ridx, sample_len)

    return X
```

Sınıflandırma işleminin daha iyi olabilmesi için ses dosyaları çoğaltılır.

```
def dataset_gen(is_train=True, batch_shape=(20, 16000), sample_augmentation=0):
    s_per_batch = batch_shape[0]
    s_len = batch_shape[1]

    X_cat = dataset['train_cat'] if is_train else dataset['test_cat']
    X_dog = dataset['train_dog'] if is_train else dataset['test_dog']

    # Go through all the permutations
    y_batch = np.zeros(s_per_batch)
    X_batch = np.zeros(batch_shape)
    # Random permutations (for X indexes)
    nbatch = int(max(len(X_cat), len(X_cat)) / s_len)
    perms = [list(enumerate([i] * nbatch)) for i in range(2)]
    perms = sum(perms, [])
    random.shuffle(perms)

    while len(perms) > s_per_batch:

        # Generate a batch
        for bidx in range(s_per_batch):
            perm, _y = perms.pop()  # Load the permutation
            y_batch[bidx] = _y

            # Select wether the sample is a cat or a dog
            _X = X_cat if _y == 0 else X_dog

            # Apply the permutation to the good set
            if is_train:
                X_batch[bidx] = get_augmented_trunk(
                    _X,
                    idx=perm,
                    sample_len=s_len,
                    added_samples=sample_augmentation)
            else:
                X_batch[bidx] = get_trunk(_X, perm, s_len)

        yield (X_batch.reshape(s_per_batch, s_len, 1),
               y_batch.reshape(-1, 1))
```

Veriseti hazırlanır.

```
def load_dataset(dataframe):
    df = dataframe

    dataset = {}
    for k in ['train_cat', 'train_dog', 'test_cat', 'test_dog']:
        v = list(df[k].dropna())
        v = read_wav_files(v)
        v = np.concatenate(v).astype('float32')

        # Compute mean and variance
        if k == 'train_cat':
            dog_std = dog_mean = 0
            cat_std, cat_mean = v.std(), v.mean()
        elif k == 'train_dog':
            dog_std, dog_mean = v.std(), v.mean()

        # Mean and variance suppression
        std, mean = (cat_std, cat_mean) if 'cat' in k else (dog_std, dog_mean)
        v = (v - mean) / std
        dataset[k] = v

        print('loaded {} with {} sec of audio'.format(k, len(v) / 16000))

    return dataset

df = pd.read_csv(CSV_PATH)
dataset = load_dataset(df)
```

Gerekli sabitler ayarlanır ve ağ modeli belirlenir.

```
batch_size=512
num_data_points = 16000
n_augment = 10

from tensorflow.keras import backend as K
K.clear_session()

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import Embedding, BatchNormalization
from tensorflow.keras.layers import Conv1D, GlobalAveragePooling1D, MaxPooling1D
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Conv1D(20, 4, strides=2, activation='relu', input_shape=(num_data_points, 1)))
model.add(BatchNormalization())
model.add(Conv1D(20, 4, strides=2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(2))
model.add(Conv1D(40, 4, strides=2, activation='relu'))
model.add(BatchNormalization())
model.add(Conv1D(40, 4, strides=2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(2))
model.add(Conv1D(80, 4, strides=2, activation='relu'))
model.add(BatchNormalization())
model.add(Conv1D(80, 4, strides=2, activation='relu'))
model.add(BatchNormalization())
model.add(GlobalAveragePooling1D())
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

Model derlenir ve eğitim işlemi gerçekleştirilir.

```
NUM_EPOCHS = 50
adam_optimizer = Adam(decay=1e-3)
model.compile(loss='binary_crossentropy',
              optimizer=adam_optimizer,
              metrics=['accuracy'])

NUM_EPOCHS = 50

train_loss = []
val_loss = []
train_acc = []
val_acc = []

# Loop through epoch samples (batchs)
for epochs in range(NUM_EPOCHS):
    train_gen = dataset_gen(is_train=True, batch_shape=(batch_size, num_data_points), sample_augmentation=n_augment)

    for batch_x, batch_y in train_gen:
        history = model.fit(batch_x, batch_y, epochs=1, validation_split=0.2)
        train_loss.extend(history.history['loss'])
        val_loss.extend(history.history['val_loss'])
        train_acc.extend(history.history['accuracy'])
        val_acc.extend(history.history['val_accuracy'])
```

Sistemin başarımı ve kayıpları grafik üzerinde gösterilir.

```
fig = plt.figure(figsize=(15,8))
ax = fig.add_subplot(111)
ax.plot(train_loss, label="train loss")
ax.plot(val_loss, label="val loss", color='green')
plt.legend()
plt.title("Log Loss")
plt.show()

fig = plt.figure(figsize=(15,8))
ax = fig.add_subplot(111)
ax.plot(train_acc, label="training accuracy")
ax.plot(val_acc, label="val accuracy", color='green')
plt.legend()
plt.title("Accuracy")
plt.show()
```

Uygulama 7. Değerlendirme Soruları

1) Verilen kodu inceleyerek kendi modelinizi oluşturunuz **(10p).**

2) Kalp verisetine ait verileri kullanarak bir ESA mimarisi geliştiriniz. Geliştirmiş olduğunuz modelin doğrulama (accuracy), F1-skor ve kayıp değerlerini bulunuz. Bulduğunuz bu değerleri grafik üzerinde gösteriniz. Bunlara ek olarak tahmin işlemi gerçekleştiriniz **(45p)**.

3) Çevre verisetine ait verileri kullanarak bir ESA mimarisi geliştiriniz. Geliştirmiş olduğunuz modelin doğrulama (accuracy), F1-skor ve kayıp değerlerini bulunuz. Bulduğunuz bu değerleri grafik üzerinde gösteriniz. Bunlara ek olarak karmaşıklık matrisi oluşturunuz **(45p)**.