Cat's Breed Dataset

İrem Çakmak

203908028

Kedi ırklarını sınıflandırmak için bir makine öğrenimi modeli oluşturmak için gerekli verileri hazırlar. Bunun için öncelikle belirtilen klasördeki verileri kullanır. Kedi ırkları belirlenir ve resim boyutları ayarlanır. Sonrasında, veri ve etiket listeleri oluşturulur ve veri setindeki resimler bu listelere eklenir. Bu işlem verilerin yüklenmesi ve işlenmesi için kullanılacaktır.

In [1]:
```
import numpy as np
import os
import cv2
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D, Activation
```

In [12]:
```
# Veri yolu

data_path = "cat_v1/"
classes = {"bengal": 0, "domestic_shorthair": 1,"maine_coon":2,"ragdoll":3 ,"siamese":4}
num_classes = len(classes)
```

In [13]:
```

# Resim boyutu
img_size = 64
```

In [14]:
```
# Veri ve etiket listeleri
data = []
labels = []
```

Veri setindeki resimleri ve onların etiketlerini numpy dizilerine dönüştürür. Resimler boyutlandırılır ve veri setindeki her bir resim, onların etiketleri ile birlikte "data" ve "labels" listelerine eklenir. Veriler eğitim ve test setleri olarak ayrılır ve etiketler ikili sınıf matrisine dönüştürülür.

In [15]:
```
# Resimleri ve etiketleri listele

for folder, cl in classes.items():
    folder_path = os.path.join(data_path, folder)
    for img_name in os.listdir(folder_path):
        img_path = os.path.join(folder_path, img_name).encode('utf-8')
        img = cv2.imread(img_path.decode('utf-8'))
        if img is not None:
            img = cv2.resize(img, (img_size, img_size))
            data.append(img)
            labels.append(cl)
```

In [16]:
```
# Verileri ve etiketleri numpy dizisine dönüştür
data = np.array(data)
labels = np.array(labels)
```

In [17]:
```
# Verileri eğitim ve test setleri olarak ayı
train_data, test_data, train_labels, test_labels = train_test_split(data, labels, test_size=0.3, random_state=20)
```

In [18]:
```
# Etiketleri ikili sınıf matrisine dönüştür
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

Bu kod bloğu bir görüntü sınıflandırma modeli oluşturur ve eğitir. "cat_v1/" klasöründeki beş kedi cinsine ait görüntüler kullanılarak, Convolutional Neural Network (CNN) modeli eğitilir. Modelin doğruluğu (accuracy) ve kaybı (loss) ölçülür ve 10 epoch boyunca eğitilir. Sonuç olarak, modelin doğruluğu %85, kaybı ise 0.36'dır. Ancak, doğruluk ve kayıp, eğitim verilerine göre oldukça iyi olsa da, test verilerine göre daha düşüktür (%29). Bu sonuçlar, modelin overfitting yaptığını gösterir, yani model eğitim verilerine aşırı uyum sağlamıştır ve test verilerinde iyi performans gösterememiştir. Modelin daha iyi performans göstermesi için, overfitting'i azaltmak için farklı önlemler alınabilir.

Bu kod bloğunda, "categorical_crossentropy" kaybı ve "adam" optimizer kullanılmıştır ve "accuracy" metriği doğruluğu ölçmek için kullanılmıştır. Eğitim süreci, çoklu işlem yapısı kullanılarak hızlandırılmıştır. Eğitim sonunda, modelin test verileri üzerindeki performansı hesaplanır ve test kayıp değeri 3.57, doğruluk oranı ise %30 olarak raporlanır.

In [19]:
```
# Model oluşturma

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(img_size, img_size, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, kernel_size=(3, 3)))
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax'))
```

In [20]:
```
# Model derleme
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [21]:
```
# Model eğitimi
model.fit(train_data, train_labels, batch_size=32, epochs=10, verbose=1, validation_data=(test_data, test_labels), use_multiproce
```

```
Epoch 1/10
21/21 [==============================] - 5s 193ms/step - loss: 293.5701 - accuracy: 0.2009 - val_loss: 1.6807 - val_accuracy:
0.2238
Epoch 2/10
21/21 [==============================] - 4s 191ms/step - loss: 2.0792 - accuracy: 0.2924 - val_loss: 1.6202 - val_accuracy: 0.2
867
Epoch 3/10
21/21 [==============================] - 4s 191ms/step - loss: 1.4049 - accuracy: 0.4063 - val_loss: 1.6234 - val_accuracy: 0.2
937
Epoch 4/10
21/21 [==============================] - 4s 203ms/step - loss: 1.2021 - accuracy: 0.5022 - val_loss: 1.6214 - val_accuracy: 0.2
832
Epoch 5/10
21/21 [==============================] - 4s 192ms/step - loss: 1.0300 - accuracy: 0.5562 - val_loss: 1.6340 - val_accuracy: 0.2
832
Epoch 6/10
21/21 [==============================] - 4s 188ms/step - loss: 0.8134 - accuracy: 0.6657 - val_loss: 1.7149 - val_accuracy: 0.2
797
Epoch 7/10
21/21 [==============================] - 4s 184ms/step - loss: 0.6648 - accuracy: 0.7271 - val_loss: 1.8392 - val_accuracy: 0.2
937
Epoch 8/10
21/21 [==============================] - 4s 178ms/step - loss: 0.4563 - accuracy: 0.8156 - val_loss: 2.1055 - val_accuracy: 0.3
007
Epoch 9/10
21/21 [==============================] - 4s 184ms/step - loss: 0.4187 - accuracy: 0.8486 - val_loss: 2.1886 - val_accuracy: 0.3
112
Epoch 10/10
21/21 [==============================] - 4s 187ms/step - loss: 0.3603 - accuracy: 0.8591 - val_loss: 2.3464 - val_accuracy: 0.2
902
```

Out[21]:
```
<keras.callbacks.History at 0x1741bc07970>
```

Test veri kümesinde 2.35'te bir kayıp (loss) ve yaklaşık %29 doğruluk (accuracy) elde edildiğini göstermektedir.

In [22]:
```
# Model değerlendirme

score = model.evaluate(test_data, test_labels, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

model = Sequential()
```

```
Test loss: 2.346421957015991
Test accuracy: 0.2902098000049591
```

Kedi resimlerinin bir veri seti üzerinde eğitilmiş bir sinir ağı modelidir ve görüntü sınıflandırma görevleri için yaygın olarak kullanılan bir tür sinir ağı olan evrişimli sinir ağı (CNN) mimarisini

kullanmaktadır. Model, 10 epoch boyunca eğitilmiştir. Sonuçlar, doğruluk oranının en yüksek %34,6 ile son epoch'ta elde edildiğini göstermektedir. Bununla birlikte, modelin eğitim doğruluğu son epoch'ta %99,25'e ulaşmıştır. Bu sonuçlar, modelin aşırı uyuma eğilimi gösterdiğini ve daha fazla veri veya düzenleme teknikleriyle geliştirilebileceğini göstermektedir. Eğitim sonunda, modelin test verileri üzerindeki performansı hesaplanır ve test kayıp değeri 3.57, doğruluk oranı ise %30 olarak raporlanır.

In [24]:
```
# 32 filtreli, 3x3 çekirdek boyutlu, relu aktivasyon fonksiyonlu ve giriş şekli (resim_yükseklik, resim_genişlik, renk_kanalları)
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 3)))
```

In [25]:
```
# 2x2 havuz boyutlu bir maksimum havuzlama katmanı ekle.
model.add(MaxPooling2D(pool_size=(2, 2)))
```

In [26]:
```
#64 filtreli ve 3x3 çekirdek boyutlu başka bir evrişim katmanı ekle.
model.add(Conv2D(64, (3, 3), activation='relu'))
```

In [27]:
```
# 2x2 boyutunda bir maksimum havuzlama katmanı ekle.
model.add(MaxPooling2D(pool_size=(2, 2)))
```

In [28]:
```
# Önceki katmanın çıktısını düzleştirin.
model.add(Flatten())
```

In [29]:
```
# 128 nöronlu ve relu aktivasyon fonksiyonlu tam bağlı bir katman ekle.
model.add(Dense(128, activation='relu'))
```

In [30]:
```
# Son çıktı katmanı olarak, softmax aktivasyon fonksiyonu ile bir çıktı katmanı ekleyin (nöron sayısı sınıf sayısıyla eşit olmalı
model.add(Dense(num_classes, activation='softmax'))
```

In [31]:
```
#Modeli kategorik çapraz entropi kaybı fonksiyonu ve adam optimizer ile derleyin.
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [32]:
```
# Modeli, 32 öğe/batch boyutu ve 10 epoch ile eğitin:
model.fit(train_data, train_labels, batch_size=32, epochs=10, verbose=1, validation_data=(test_data, test_labels), use_multiproce
```

```
Epoch 1/10
21/21 [==============================] - 2s 81ms/step - loss: 68.9590 - accuracy: 0.2249 - val_loss: 1.7138 - val_accuracy: 0.2
378
Epoch 2/10
21/21 [==============================] - 1s 67ms/step - loss: 1.4025 - accuracy: 0.3898 - val_loss: 1.7285 - val_accuracy: 0.24
83
Epoch 3/10
21/21 [==============================] - 2s 75ms/step - loss: 0.8890 - accuracy: 0.6612 - val_loss: 1.7486 - val_accuracy: 0.30
07
Epoch 4/10
21/21 [==============================] - 1s 70ms/step - loss: 0.4699 - accuracy: 0.8381 - val_loss: 2.0455 - val_accuracy: 0.29
72
Epoch 5/10
21/21 [==============================] - 1s 69ms/step - loss: 0.3049 - accuracy: 0.9205 - val_loss: 2.3813 - val_accuracy: 0.31
12
Epoch 6/10
21/21 [==============================] - 2s 72ms/step - loss: 0.1548 - accuracy: 0.9580 - val_loss: 2.7970 - val_accuracy: 0.32
52
Epoch 7/10
21/21 [==============================] - 1s 71ms/step - loss: 0.0874 - accuracy: 0.9775 - val_loss: 3.0698 - val_accuracy: 0.34
62
Epoch 8/10
21/21 [==============================] - 1s 67ms/step - loss: 0.0866 - accuracy: 0.9820 - val_loss: 3.0128 - val_accuracy: 0.30
07
Epoch 9/10
21/21 [==============================] - 1s 67ms/step - loss: 0.0423 - accuracy: 0.9880 - val_loss: 3.2562 - val_accuracy: 0.31
12
Epoch 10/10
21/21 [==============================] - 1s 65ms/step - loss: 0.0228 - accuracy: 0.9925 - val_loss: 3.5723 - val_accuracy: 0.30
07
```

Out[32]:
```
<keras.callbacks.History at 0x1741f7f1d50>
```

In [33]:
```
score = model.evaluate(test_data, test_labels, verbose=0)
```

In [34]:
```
print('Test loss:', score[0])
```

```
Test loss: 3.572331666946411
```

In [35]:
```
print('Test accuracy:', score[1])
```

```
Test accuracy: 0.30069929361343384
```