Yüksek Düzey Programlama Dönem Proje Ödevi

Eyüp Alperen Kalkan – 202113171054 – Bilgisayar Mühendisliği – 4. Sınıf – 1. Öğretim

1. Veri Yükleme ve Dosya İşlemleri

İlk olarak verilerin bulunduğu dosyalar yükleniyor ve ardından bu dosyalar içerisinden bir zip dosyasının açılması sağlanıyor.

```
import os
from zipfile import ZipFile
file='/kaggle/input/platesv2/plates.zip'
with ZipFile(file,'r') as zip:
    zip.extractall('/kaggle/working/')
```

Bu kod, zip dosyasını açarak içindeki tüm dosyaları geçici çalışma dizinine çıkarıyor. Bu sayede veriler üzerinde işlem yapabilmemiz sağlanıyor.

2. Veri Kümesinin Yapısı

Çıkartılan veriler plates/ dizininde yer alıyor. Burada eğitim ve test verileri bulunmaktadır. Eğitim verisi daha fazla örnek içeriyor.

```
data_root = '/kaggle/working/plates/'
print(os.listdir(data_root))
```

Bu kısımda, verilerin düzenlendiği dizin yapısı kontrol ediliyor.

3. Resim Augmentasyonu (Veri Artırma)

Veri seti sınırlı olduğu için, modelin daha sağlam öğrenmesi adına veri artırma teknikleri (augmentation) uygulanıyor. Bu teknikler, resimleri döndürme, kaydırma, yansıma gibi işlemlerle çeşitlendirir.

```
train_datagen = ImageDataGenerator(validation_split=0.2,
                                   rescale=1./255,
                                   rotation_range=45,
                                   shear_range=0.2,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
```

Bu kodda yapılanlar:

- **rotation_range=45**: Resimleri rastgele 45 derece döndürüyor.
- **width_shift_range=0.2** ve **height_shift_range=0.2**: Resimleri yatay ve dikey olarak %20 kadar kaydırabiliyor.
- **zoom_range=0.2**: Resimleri %20 oranında yakınlaştırıyor ya da uzaklaştırıyor.
- **horizontal_flip=True**: Resimleri yatayda çeviriyor.

4. Veri Yükleme

Veri, flow_from_directory() fonksiyonu ile yükleniyor. Bu fonksiyon, belirtilen dizinlerden (eğitim, doğrulama, test) resimleri okur ve etiketleri (cleaned/dirty) belirler.

```
train_generator = train_datagen.flow_from_directory(directory=path_train,
                                                    classes=['cleaned','dirty'],
                                                    batch_size=64,
                                                    class_mode='binary',
                                                    shuffle=True,
                                                    target_size=(224,224),
                                                    subset='training')

validation_generator = validation_datagen.flow_from_directory(directory=path_train,
                                                              classes=['cleaned','dirty'],
                                                              batch_size=64,
                                                              class_mode='binary',
                                                              shuffle=True,
                                                              target_size=(224,224),
                                                              subset='validation')
```

Eğitim ve doğrulama verisi burada iki kategoriye (temiz ve kirli) ayrılıyor. Resimler 224x224 piksel boyutlarına yeniden boyutlandırılıyor.

5. VGG16 Modeli Kullanımı

Model olarak, önceden eğitilmiş olan **VGG16** ağını kullanıyoruz. VGG16, derin öğrenme alanında yaygın olarak kullanılan bir modeldir ve özellikle görüntü sınıflandırma için etkilidir. Ancak, burada modelin son katmanları yerine sadece özellik çıkarıcı kısmı (features extractor) kullanılacak.

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

basemodel = VGG16(include_top=False, weights='imagenet', input_shape=(224,224,3))

for layer in basemodel.layers:
    layer.trainable = False
```

- **include_top=False**: Son katmanlar (yani sınıflandırma katmanları) hariç VGG16 modelinin önceden eğitilmiş kısmı kullanılır.
- **trainable=False**: Önceden eğitilmiş katmanlar sabit tutulur ve sadece eklediğimiz katmanlar öğrenir.

6. Modelin Oluşturulması ve Eğitim

Model, VGG16 temel alınarak inşa ediliyor. Ardından bir **Dense** katmanı ekleniyor ve son olarak **sigmoid** aktivasyonu ile sınıflandırma yapılacak.

```
model = Sequential()
model.add(basemodel)
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)))
model.add(Dropout(0.6))
model.add(BatchNormalization())
model.add(Dense(1, activation='sigmoid'))
```

- **Flatten**: VGG16 modelinden alınan özellik haritaları tek boyutlu hale getiriliyor.
- **Dense(512)**: 512 nöronlu bir tam bağlantılı katman ekleniyor.
- **Dropout(0.6)**: Overfitting (aşırı öğrenme) engellemek için %60 oranında rastgele nöronlar kapatılıyor.
- **BatchNormalization**: Öğrenmeyi hızlandırmak ve stabilize etmek için batch normalization uygulanıyor.
- **Dense(1, activation='sigmoid')**: Son sınıflandırma katmanı ekleniyor ve model binary (ikili) sınıflandırma yapacak.

7. Modeli Eğitme

Modeli eğitmek için fit fonksiyonu kullanılır ve eğitim sürecinde erken durdurma (early stopping) ve öğrenme oranı azaltma (ReduceLROnPlateau) gibi callback'ler kullanılır.

```
earlystopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=1e-6)
history = model.fit(train_generator, epochs=100, validation_data=validation_generator,
                    callbacks=[earlystopping, reduce_lr], verbose=1)
```

- **monitor='val_loss'**: Validasyon kaybını izliyoruz.
- **patience=3**: 3 epoch boyunca iyileşme görülmezse eğitim durdurulur.
- **factor=0.5**: Öğrenme oranı %50 oranında azaltılır.

Eğitim başladığında, model her epoch'ta eğitilir ve doğrulama kaybı ile doğrulama doğruluğu hesaplanır.

8. Eğitim Sonuçları

Eğitim süreci boyunca doğruluk ve kayıp değerleri takip edilerek modelin performansı gözlemlenir. Ancak burada ilk başlarda doğruluk oldukça düşük ve model iyileşme gösterse de genelde sabit kalıyor.

9. Sonuçlar ve İyileştirme

Modelin doğruluğu ve kaybı iyileştirilse de, bazen doğrulama doğruluğunun arttığı ama kaybın aynı kaldığı durumlar görülüyor. Bu durum, modelin yeterince güçlü olmadığı veya veri setinin dengesiz olduğu anlamına gelebilir.

Genel olarak, modelin daha iyi sonuçlar vermesi için:

- Veri setinin büyütülmesi,
- Modelin daha fazla eğitim verisi ile beslenmesi,
- Hiperparametre ayarlarının optimize edilmesi gerekebilir.