Bilişim Tasarım Projesi Rapor

Giysi Kategorisi Tahmin Eden Derin Ağ Modeli

Haluk Esad Sarıkaya 200001680

1- Veri kümesi oluşturma

Veri kategorileri olarak tişört, sweatshirt ve pantolon olarak belirlendi. Yaklaşık 4.000 görsellik veri, Google Chrome eklentisi “download all images” kullanılarak, Yandex görseller sitesinden elde edildi.

Klasörler içinde bulunan dosyalardan .jpg uzantılı dosyalar kullanılmıştır. Bu işlem için glob modülü kullanılmıştır.

```
train_tsirts = glob.glob(train_path+"/tsirts/*.jpg")
train_sweatshirt = glob.glob(train_path+"/sweatshirt/*.jpg")
train_pants = glob.glob(train_path+"/pants/*.jpg")
```

Görseller, liste haline getirilmiş ve ardından “pandas” kütüphanesi yardımıyla satır ve sütun tablosu haline getirilmiştir (0, 1 ve 2 etiketleri sırasıyla tişört, sweatshirt ve pantolon sınıflarını işaret etmektedir).

	label	image
0	0	data/tsirts\-.jpg
1	0	data/tsirts\-2.jpg
2	0	data/tsirts\-com-fruit-of-the-loom-sleeve-t-shirts-5abde166662934.3573392215223934464185.jpg
3	0	data/tsirts\-din-t-shirt-med-dit-egen-oploadet-bil-5b67d9f982e579.13529464153535326655362.jpg
4	0	data/tsirts\-faintest-idea-flying-the-japanese-flag-png-favpng-PJWP5wrwWfYCC3B1A9CrwTPPd.jpg
...	...	...
3967	2	data/pants_57.JPG2.jpg
3968	2	data/pants_57.JPG3.jpg
3969	2	data/pants_57.JPG4.jpg
3970	2	data/pants_57.JPG6.jpg

2- Önişleme

Görüntüler, belli bir biçime dönüştürülüp ardından modele aktarılmalıdır. Bunun için NumPy dizini kullanılır.

Listedeki bütün resimler 128x128 boyutuna ve RGB renklerine dönüştürülür. Ardından elemanlar NumPy dizinine taşınır.

```
path = ''
img_list = list(df_train['image'])
data_img = [] #oluşturulan array adı
for each in img_list:
    each_path = os.path.join(path, each)# 3 dosya yolu için
    print(each_path)
    each_img = cv2.imread(each_path) # her resim için
    each_img = cv2.cvtColor(each_img, cv2.COLOR_BGR2RGB) # rgb renklerine dönüştürme (resimler 3 depth kazanır)
    each_img_resized = cv2.resize(each_img, (128,128)) # resimlerin boyutu 128x128
    data_img.append(each_img_resized) # boyutları ve renkleri değişmiş bütün resimler liste haline getirilir

X = np.array(data_img) # oluşturulan liste NumPy array'e dönüştürülür.
```

Oluşturulan NumPy dizininde 3972 adet resim bulunuyor ve resimler 128x128 boyutunda olup 3 derinliğe sahip.

```
Shape of X:  (3972, 128, 128, 3)
```

Ayrıca resimlerin etiketlerini ayrı bir dizinde saklamamız gerekiyor. Scikit-learn kütüphanesinin altında, veri ön hazırlaması aşamasında kullanılan OneHotEncoder adı verilen encoder kullanılmıştır. Etiketleri matris haline getirir ve makine öğrenmesi algoritmasında verim sağlar.

```
y = OneHotEncoder(dtype='int8', sparse=False).fit_transform(df_train['label'].values.reshape(-1,1))
```

```
print('Shape of y: ', y.shape)
```

```
Shape of y:  (3972, 3)
```

3- Görüntü Sınıflandırma

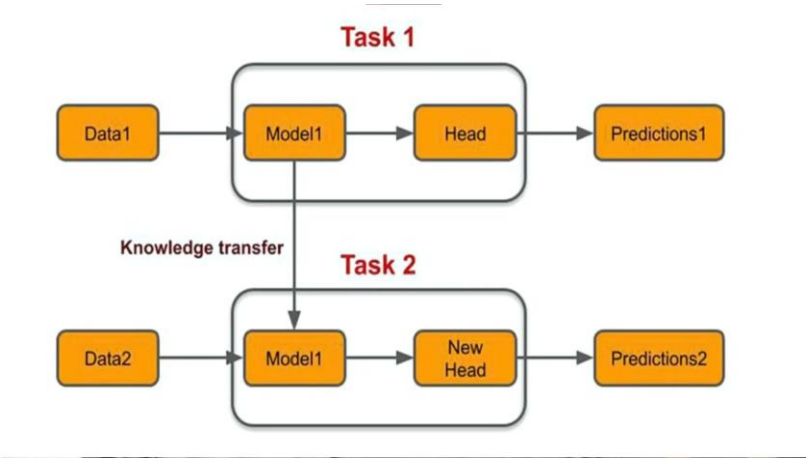
Bu aşamada, kullandığımız veri setini bölmemiz gerekmektedir. Böylelikle eğitim seti üzerinde modeli eğitir ve test seti üzerinde tahminler yapılır.

```
X_data, X_test, y_data, y_test = train_test_split(X, y, test_size=0.15, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_data, y_data, test_size=0.2, random_state=42)
print('X_train shape: ', X_train.shape)
print('y_train shape: ', y_train.shape)
print('X_val shape  : ', X_val.shape)
print('y_val shape  : ', y_val.shape)
print('X_test shape : ', X_test.shape)
print('y_test shape : ', y_test.shape)
#
```

```
X_train shape:  (2700, 128, 128, 3)
y_train shape:  (2700, 3)
X_val shape  :  (676, 128, 128, 3)
y_val shape  :  (676, 3)
X_test shape :  (596, 128, 128, 3)
y_test shape :  (596, 3)
```

Böylece toplamda 6 adet NumPy dizini elde edilmiş olur.

Projenin bu noktasında pre-trained model kullanılmıştır. Kullanılan pre-trained modeli, Xception'dır. Google tarafından geliştirilip ImageNet veri setleri kullanılarak eğitilmiştir.



Daha önce eğitilen verileri kullanılarak sonuca ulaşılmıştır.

Xception tanımlama:

```
base = Xception(include_top=False,
                weights='imagenet',
                input_shape=(128,128,3))
x = base.output
x = GlobalAveragePooling2D()(x) # 2B a dönüştürür
```

Projede toplam 3 class bulunduğundan softmax aktivasyonu, loss function olarak ise categorical crossentropy kullanılmıştır.

```
head = Dense(3, activation='softmax')(x)
model = Model(inputs=base.input, outputs=head)
```

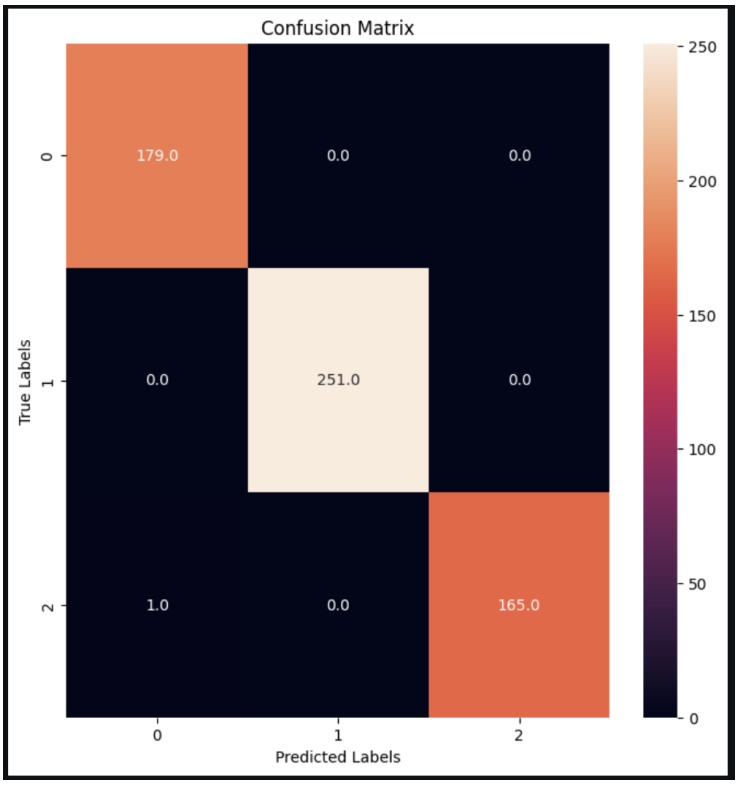
Model eğitimi aşağıda verilen görseldeki gibi yapılmıştır.

```
history = model.fit_generator(
              train_gen.flow(X_train, y_train,
              batch_size=batch_size),
              epochs = epochs,
              validation_data = validation_gen.flow(X_val, y_val)
)
```

```
Epoch 1/4
54/54 [==============================] - 176s 3s/step - loss: 0.4766 - accuracy: 0.8496 - val_loss: 0.2740 - val_accuracy: 0.88
31
Epoch 2/4
54/54 [==============================] - 175s 3s/step - loss: 0.0734 - accuracy: 0.9822 - val_loss: 0.0446 - val_accuracy: 0.98
37
Epoch 3/4
54/54 [==============================] - 175s 3s/step - loss: 0.0310 - accuracy: 0.9900 - val_loss: 0.0150 - val_accuracy: 0.99
70
Epoch 4/4
54/54 [==============================] - 176s 3s/step - loss: 0.0160 - accuracy: 0.9948 - val_loss: 0.0070 - val_accuracy: 0.99
85
```

4- Sonuçları Analiz Etme

Aşağıdaki confusion matriste görüldüğü üzere neredeyse düzgün çalışmaktadır. Sadece pantolon sınıfında küçük bir hata yapıldığı görülmektedir.



Aşağıda kayıp ve doğruluk fonksiyonunun grafiği görülmektedir.